北 京 林 业 大 学

2019 学年—2020 学年第 2 学期数字图像处理实验报告书

专业：计算机科学与技术(创新实验班)　班级：计创 18

姓　　名：连月菡　　学　　号：181002222

实验地点：家　　任课教师：王忠芝

实验题目：实验 3 图像处理程序设计

实验环境：Visual Studio 2019 Community

一、 实验目的

1、 利用 c/c++ 作为编程环境进行图像处理的编程实践，掌握 Windows 下的 DIB 图像的存储格式和编程方法。

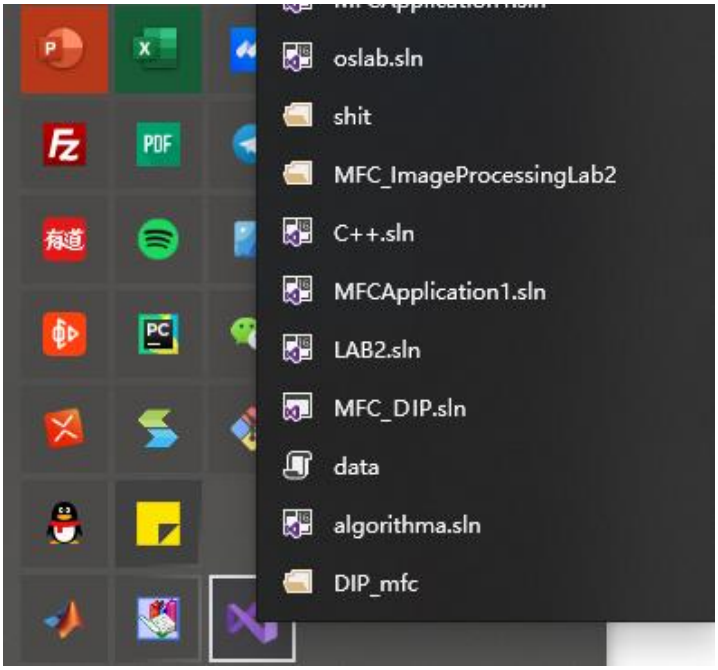
2、 掌握图像线性变换（图像求反等）的处理方法。

二、 实验内容

1. 编写程序处理 BMP 格式的图像。
2. 掌握图像线性变换的处理方法。

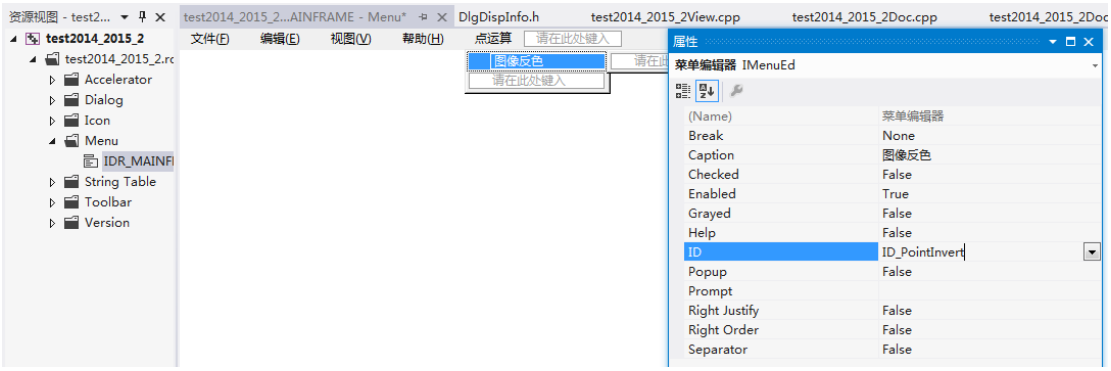
三、 实验步骤

1、 打开上次实验的工程,继续对已有的图像程序进行功能的添加。

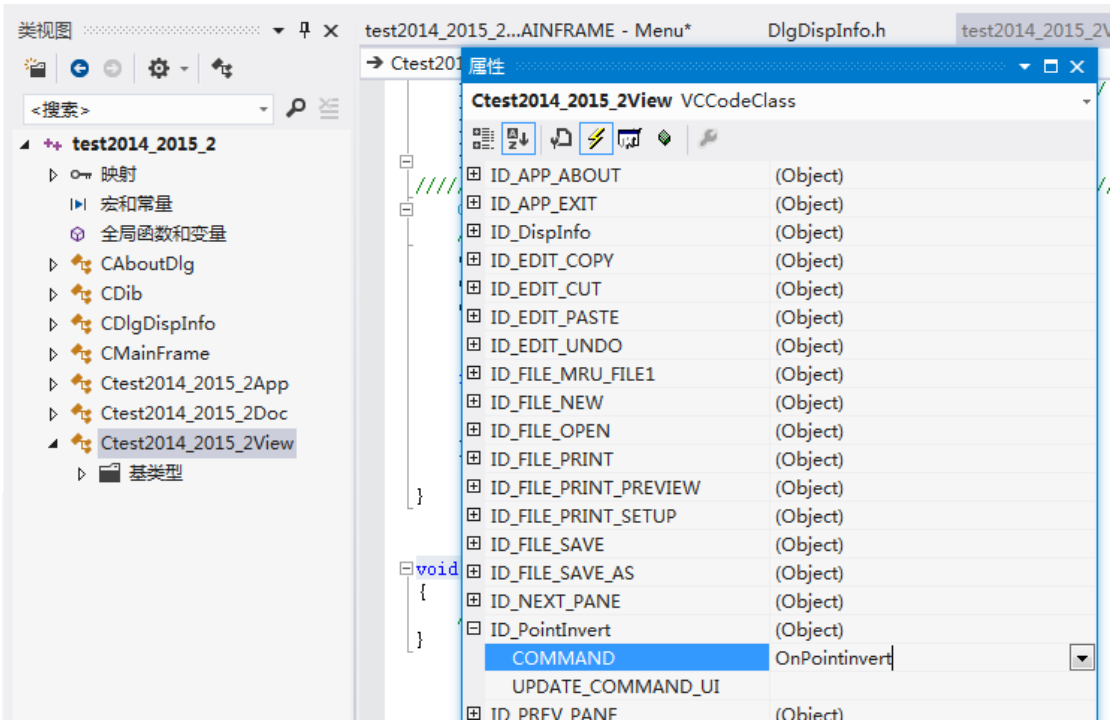


2、　线性变换和分段线性变换

1、添加“图像反色”功能。首先添加一个“点运算”的菜单，再添加一个“图像反色”的子菜单。**定义 ID。**



2、在 View 类中添加菜单项的处理程序。



3、程序代码。

```
1.  void CMFCApplication1View::OnPintinvert()
2.  {
3.      // TODO: 在此添加命令处理程序代码
4.      //图像反色
5.      CMFCApplication1Doc* pDoc = GetDocument();
6.      long lSrcLineBytes;
7.      long lSrcWidth;
8.      long lSrcHeight;
9.      LPSTR lpSrcDib;
```

```
    LPSTR lpSrcStartBits;
    lpSrcDib = (LPSTR)::GlobalLock((HGLOBAL)pDoc->GetHObject());
    if (!lpSrcDib) return;
    if (pDoc->m_dib.GetColorNum(lpSrcDib) != 256)
    {
        AfxMessageBox(L"对不起,不是 256 色位图!");
        ::GlobalUnlock((HGLOBAL)pDoc->GetHObject());
        return;
    }
    lpSrcStartBits = pDoc->m_dib.GetBits(lpSrcDib);
    lSrcWidth = pDoc->m_dib.GetWidth(lpSrcDib);
    lSrcHeight = pDoc->m_dib.GetHeight(lpSrcDib);
    lSrcLineBytes = pDoc->m_dib.GetReqByteWidth(lSrcWidth * 8);
    FLOAT fA = -1.0;
    FLOAT fB = 255;
    LinerTrans(lpSrcStartBits, lSrcWidth, lSrcHeight, fA, fB);
    pDoc->SetModifiedFlag(TRUE);
    pDoc->UpdateAllViews(NULL);
    ::GlobalUnlock((HGLOBAL)pDoc->GetHObject());
}
```

给对话框添加新的类

2、添加“线性变换”功能。在“点运算”的**菜单**中再添加一个“线性变换”的子菜单。**定义 ID。**

3、为“线性变换”的功能添加参数**对话框**。

对话框上放置控件。

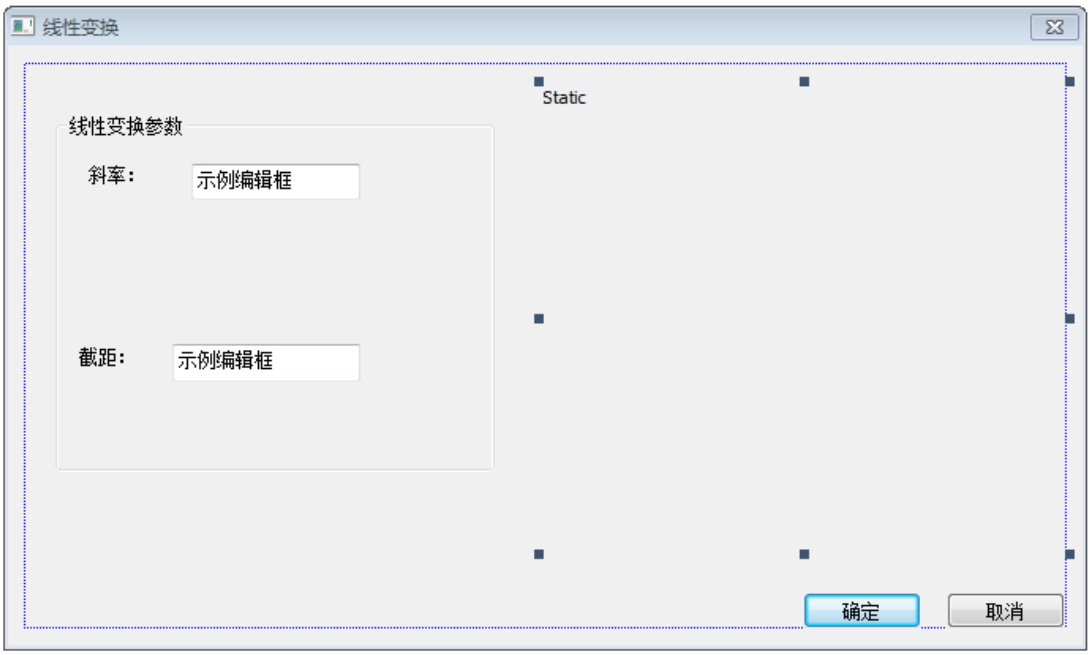
一个组控件用于框住斜率和截距。

两个静态文本控件，用于说明文字。

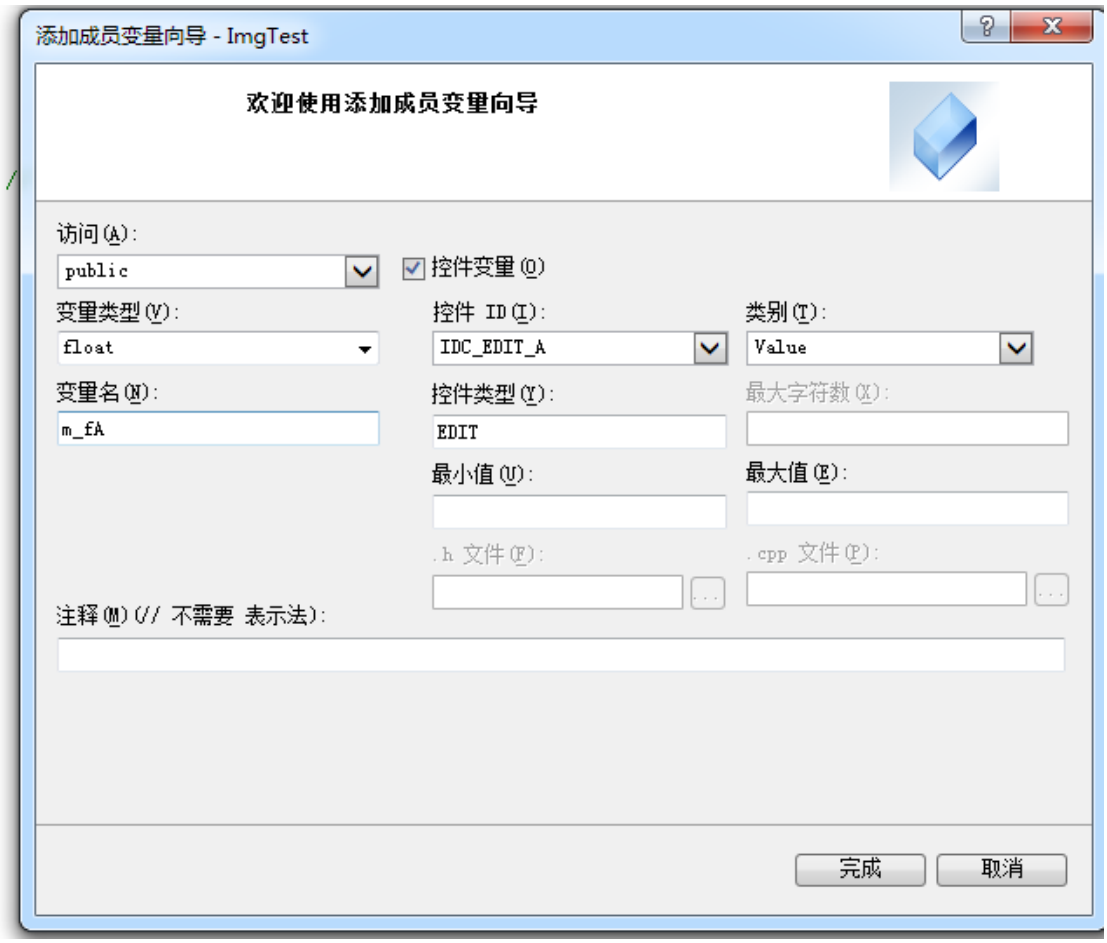
两个示例编辑框，用于输入斜率和截距。

另外一个静态文本框，用于显示线性变换曲线。

进行属性设置。定义类。为控件添加变量。



(1)属性设置：
对话框 ID：IDD_DLG_LinerPara
编辑框 ID：斜率 IDC_EDIT_A,截距：斜率 IDC_EDIT_B，
静态文本框 ID（显示变换曲线）：IDC_COORD
(2)用类向导为对话框建立类。CDlgLinerPara
(3)为对话框上的控件定义变量：
IDC_EDIT_A：float　　m_fA
IDC_EDIT_B: float　　m_fB



4、在 View 类中添加菜单项“线性变换”的处理程序。并添加代码。

并在相应的函数下添加代码

```
1. 一、调用对话框部分（VIEW 类中菜单函数，代码片段）
2.
3.     CDlgInverse_V dlgPara;// 创建对话框
4.     dlgPara.m_threshold =128;
5.     if (dlgPara.DoModal() != IDOK)// 显示对话框
6.     {
7.         return;
8.     }
9.     threshold=dlgPara.m_threshold;
10.
11. 二、对话框类
12.
13. // DlgInverse_V.cpp : 实现文件
14. //
15.
16. #include "stdafx.h"
17. #include "test2014_2015_2.h"
18. #include "DlgInverse_V.h"
```

```
#include "afxdialogex.h"


// CDlgInverse_V 对话框

IMPLEMENT_DYNAMIC(CDlgInverse_V, CDialogEx)

CDlgInverse_V::CDlgInverse_V(CWnd* pParent /*=NULL*/)
    : CDialogEx(CDlgInverse_V::IDD, pParent)
    , m_threshold(128)
{

}

CDlgInverse_V::~CDlgInverse_V()
{
}

void CDlgInverse_V::DoDataExchange(CDataExchange* pDX)
{
    CDialogEx::DoDataExchange(pDX);
    DDX_Text(pDX, IDC_Threshold, m_threshold);
}


BEGIN_MESSAGE_MAP(CDlgInverse_V, CDialogEx)
    ON_WM_PAINT()
    ON_EN_KILLFOCUS(IDC_Threshold, &CDlgInverse_V::OnEnKillfocusThreshold)
    ON_WM_LBUTTONDOWN()
    ON_WM_LBUTTONUP()
    ON_WM_MOUSEMOVE()
    ON_BN_CLICKED(IDC_RADIO1, &CDlgInverse_V::OnBnClickedRadio1)
    ON_BN_CLICKED(IDC_RADIO2, &CDlgInverse_V::OnBnClickedRadio2)
END_MESSAGE_MAP()


// CDlgInverse_V 消息处理程序


BOOL CDlgInverse_V::OnInitDialog()
{
    CDialogEx::OnInitDialog();
    //获取绘制图形的标签
    CWnd* pWnd=GetDlgItem(IDC_COORD);
```

```
63.     //计算接受鼠标事件的有效区域
64.     pWnd->GetClientRect (m_MouseRect );
65.     pWnd->ClientToScreen(&m_MouseRect );
66.
67.     CRect rect;
68.     GetClientRect(rect);
69.     ClientToScreen(&rect);
70.
71.     m_MouseRect.top-=rect.top;
72.     m_MouseRect.left -=rect.left;
73.     //设置接受鼠标事件的有效区域
74.     m_MouseRect.top+=25;
75.     m_MouseRect.left+=10;
76.     m_MouseRect .bottom =m_MouseRect .top+255;
77.     m_MouseRect.right=m_MouseRect.left +256;
78.     //初始化拖动状态
79.     m_bIsDrawing=FALSE;
80.
81.
82.     return true;
83. }
84.
85.
86. void CDlgInverse_V::OnPaint()
87. {
88.     CPaintDC dc(this); // device context for painting
89.
90.
91.     //获取绘制坐标的文本框
92.     CWnd * pWnd=GetDlgItem(IDC_COORD);
93.     //字符串
94.     CString str;
95.     CPoint pLeft,pRight,pCenterTop,pCenterBottom;
96.     //指针
97.     CDC *pDC=pWnd->GetDC();
98.     pWnd->Invalidate();
99.     pWnd->UpdateWindow();
100.      //x1(left),y1(top),x2(right),y2(bottom)
101.      pDC->Rectangle (0,0,330,300);
102.      //创建画笔对象
103.      CPen* pPenRed=new CPen;
104.      //红色画笔，红绿蓝
105.      pPenRed->CreatePen(PS_SOLID,2,RGB(255,0,0));
106.      //创建画笔对象
```

```
107.     CPen* pPenBlue=new CPen;
108.     //蓝色画笔，红绿蓝
109.     pPenBlue->CreatePen(PS_SOLID,2,RGB(0,0,255));
110.     //创建画笔对象
111.     CPen* pPenGreen=new CPen;
112.     //绿色画笔，红绿蓝
113.     pPenGreen->CreatePen(PS_DOT,1,RGB(0,255,0));
114.     //选中当前红色画笔，并保存以前的画笔
115.     CGdiObject* pOldPen=pDC->SelectObject(pPenRed);
116.     //绘制坐标轴
117.     pDC->MoveTo(10,10);
118.     //垂直轴
119.     pDC->LineTo(10,280);
120.     //水平轴
121.     pDC->LineTo(320,280);
122.     //写坐标
123.     str.Format(L"0");
124.     pDC->TextOut(10,281,str);
125.     str.Format(L"255");
126.     pDC->TextOut(265,281,str);
127.     pDC->TextOutW(11,25,str);
128.     //绘制 X 箭头
129.     pDC->LineTo(315,275);
130.     pDC->MoveTo(320,280);
131.     pDC->LineTo(315,285);
132.     //绘制 Y 箭头
133.     pDC->MoveTo(10,10);
134.     pDC->LineTo(5,15);
135.     pDC->MoveTo(10,10);
136.     pDC->LineTo(15,15);
137.     //更改成绿色画笔,画垂直线
138.
139.     pDC->SelectObject(pPenGreen);
140.     pCenterTop.x=10+m_threshold;//128;
141.     pCenterTop.y=25;
142.
143.     pCenterBottom.x=10+m_threshold;//128;
144.     pCenterBottom.y=280;
145.     pDC->MoveTo( pCenterTop);
146.     pDC->LineTo( pCenterBottom);
147.
148.     //更改成蓝色画笔
149.     pDC->SelectObject(pPenBlue);
150.     //画两条变换线，没有计算斜率,没有交互
```

```
151.     pLeft.x=10;
152.     pLeft.y=280;
153.     //pDC->MoveTo(10,280);
154.     pDC->MoveTo(pLeft);
155.     pCenterTop.x=10+m_threshold;//128;
156.     pCenterTop.y=25;
157.     //pDC->LineTo(138,25);
158.     pDC->LineTo(pCenterTop);
159.     pRight.x=265;
160.     pRight.y=280;
161.     //pDC->LineTo(265,280);
162.     pDC->LineTo(pRight );
163. 
164. 
165. 
166. 
167. }
168. 
169. 
170. void CDlgInverse_V::OnEnKillfocusThreshold()
171. {
172.     // TODO: 在此添加控件通知处理程序代码
173.     //保存用户设置
174.     UpdateData(TRUE);
175.     //重绘
176.     InvalidateRect(m_MouseRect,TRUE);
177. }
178. 
179. 
180. void CDlgInverse_V::OnLButtonDown(UINT nFlags, CPoint point)
181. {
182.     //  当用户单击鼠标左键开始拖动
183.     //判断是否在有效区域
184.     if(m_MouseRect.PtInRect (point))
185.     {
186.         if(point.x==(m_MouseRect.left +m_threshold ))
187.         {
188.             //设置拖动状态
189.             m_bIsDrawing =TRUE;
190.             //更改光标
191.             ::SetCursor(::LoadCursor(NULL,IDC_SIZEWE));
192.         }
193.     }
194. 
```

```
195.     //默认
196.     CDialogEx::OnLButtonDown(nFlags, point);
197. }
198. 
199. 
200. void CDlgInverse_V::OnLButtonUp(UINT nFlags, CPoint point)
201. {
202.     //  当用户释放鼠标左键停止拖动
203.     if(m_bIsDrawing )
204.     {
205.         //重新设置拖动状态
206.         m_bIsDrawing =FALSE;
207.     }
208.     //默认
209.     CDialogEx::OnLButtonUp(nFlags, point);
210. }
211. 
212. 
213. void CDlgInverse_V::OnMouseMove(UINT nFlags, CPoint point)
214. {
215.     //  判断当前光标是否在绘制区域
216.     if(m_MouseRect .PtInRect (point))
217.     {
218.         //判断是否在拖动
219.         if(m_bIsDrawing)
220.         {
221.             //更改阈值
222.             m_threshold =(BYTE)(point.x-m_MouseRect .left );
223.             //更改光标
224.             ::SetCursor (::LoadCursor (NULL,IDC_SIZEWE));
225.             //更新
226.             UpdateData(FALSE);
227.             //重绘
228.             InvalidateRect(m_MouseRect ,TRUE);
229.         }
230.         else if(point.x==(m_MouseRect.left+m_threshold))
231.         {
232.             //更改光标
233.             ::SetCursor (::LoadCursor (NULL,IDC_SIZEWE));
234.         }
235.     }
236.     //默认
237.     CDialogEx::OnMouseMove(nFlags, point);
238. }
```

```
239.
240.
241. void CDlgInverse_V::OnBnClickedRadio1()
242. {
243.     // TODO: 在此添加控件通知处理程序代码
244.     temp=1;
245.     m_threshold=temp;
246.     UpdateData(FALSE);
247. }
248.
249.
250. void CDlgInverse_V::OnBnClickedRadio2()
251. {
252.     // TODO: 在此添加控件通知处理程序代码
253.     temp=2;
254.     m_threshold=temp;
255.     UpdateData(FALSE);
256. }
257.
258. ----------------------------------------
259. #pragma once
260.
261.
262. // CDlgInverse_V 对话框
263.
264. class CDlgInverse_V : public CDialogEx
265. {
266.     DECLARE_DYNAMIC(CDlgInverse_V)
267.
268. public:
269.     CDlgInverse_V(CWnd* pParent = NULL);   // 标准构造函数
270.     virtual ~CDlgInverse_V();
271.     //
272.     int temp;
273.     //响应鼠标的区域
274.     CRect m_MouseRect;
275.     //标识是否拖动
276.     BOOL m_bIsDrawing;
277.     //
278.
279. // 对话框数据
280.     enum { IDD = IDD_Dlg_Inverse_V };
281.
282. protected:
```

```
283.     virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV 支持
284. 
285.     DECLARE_MESSAGE_MAP()
286. public:
287.     int m_threshold;
288.     virtual BOOL OnInitDialog();
289.     afx_msg void OnPaint();
290.     afx_msg void OnEnKillfocusThreshold();
291.     afx_msg void OnLButtonDown(UINT nFlags, CPoint point);
292.     afx_msg void OnLButtonUp(UINT nFlags, CPoint point);
293.     afx_msg void OnMouseMove(UINT nFlags, CPoint point);
294.     afx_msg void OnBnClickedRadio1();
295.     afx_msg void OnBnClickedRadio2();
296. };
```

5、 为**对话框类**添加处理函数（通过**属性**设置进行添加）

virtual BOOL OnInitDialog(); //重写函数

afx_msg void OnKillfocusEditA(); //事件函数

afx_msg void OnKillfocusEditB(); //事件函数

afx_msg void OnLButtonDown(UINT nFlags, CPoint point); //消息函数

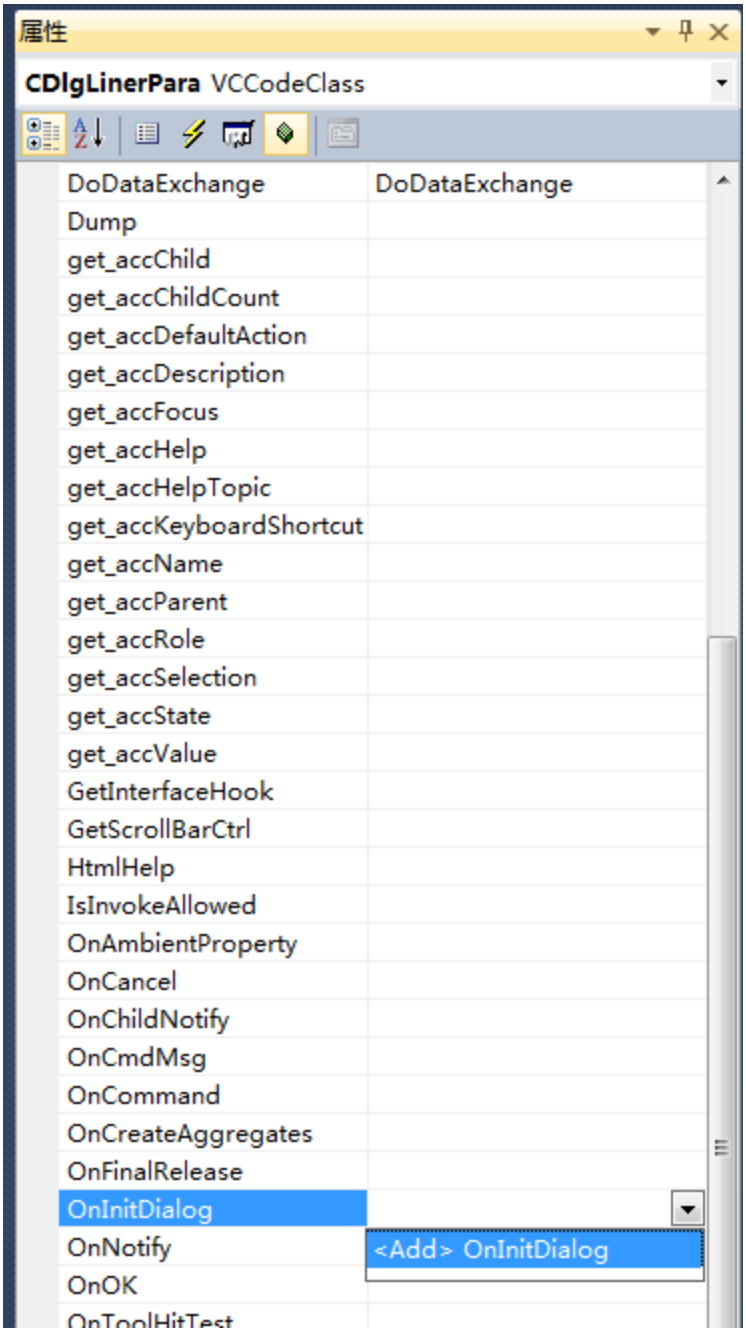
afx_msg void OnMouseMove(UINT nFlags, CPoint point); //消息函数

afx_msg void OnLButtonUp(UINT nFlags, CPoint point); //消息函数

afx_msg void OnPaint(); //消息函数

（1） 添加重写函数

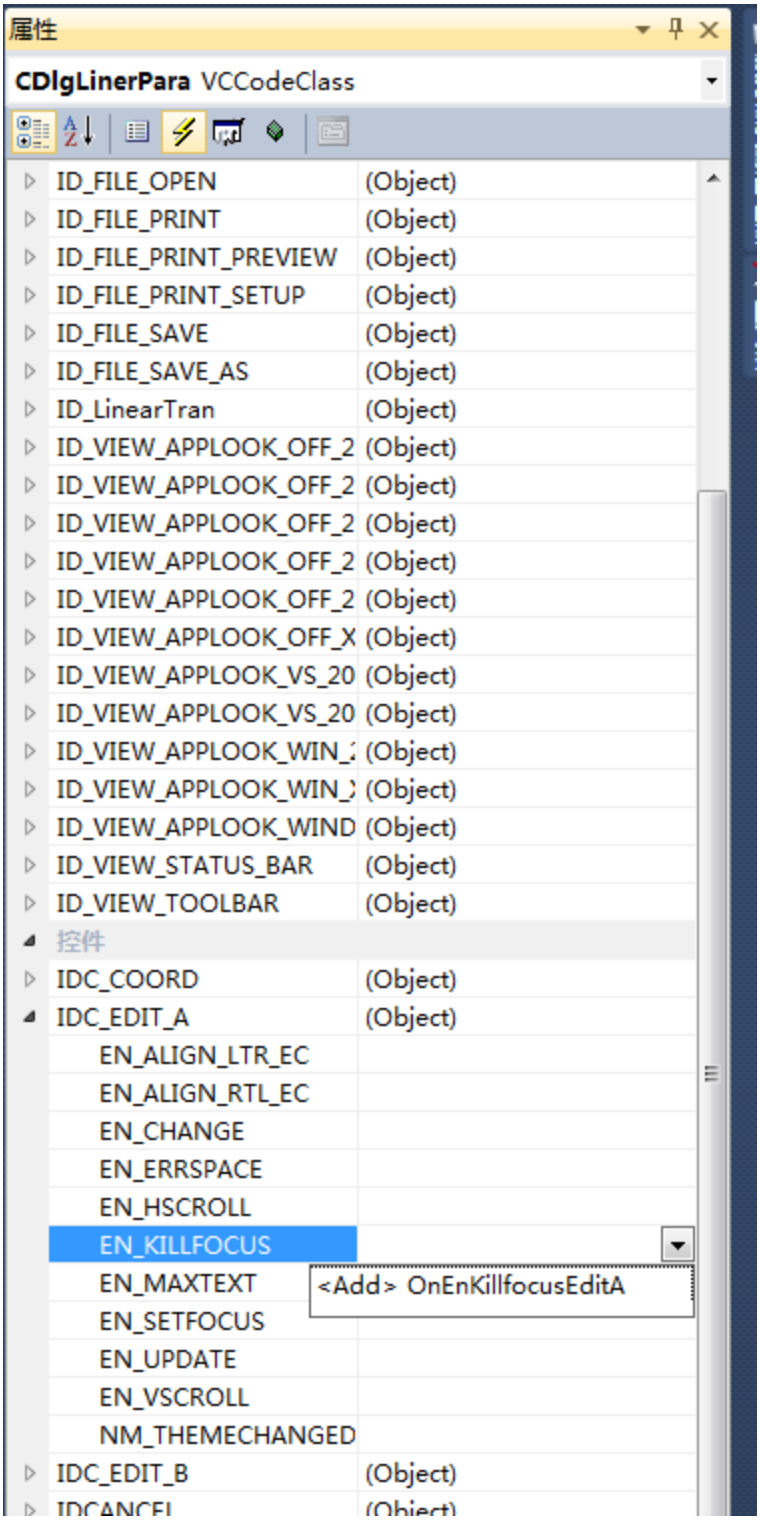
virtual BOOL OnInitDialog();

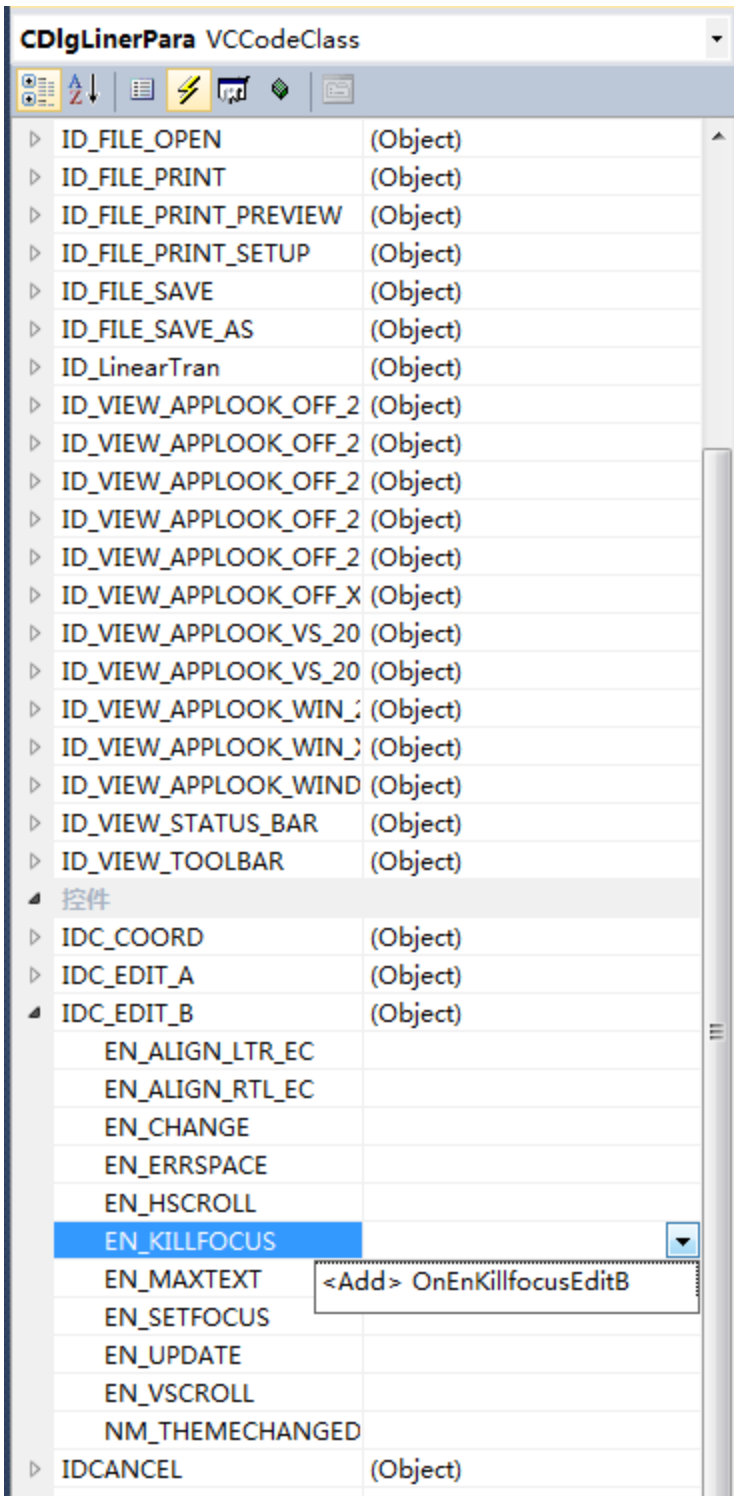


（2）添加事件函数

afx_msg void OnKillfocusEditA();

afx_msg void OnKillfocusEditB();


属性
CDlgLinerPara VCCodeClass
ID_FILE_OPEN (Object)
ID_FILE_PRINT (Object)
ID_FILE_PRINT_PREVIEW (Object)
ID_FILE_PRINT_SETUP (Object)
ID_FILE_SAVE (Object)
ID_FILE_SAVE_AS (Object)
ID_LinearTran (Object)
ID_VIEW_APPLOOK_OFF_2 (Object)
ID_VIEW_APPLOOK_OFF_2 (Object)
ID_VIEW_APPLOOK_OFF_2 (Object)
ID_VIEW_APPLOOK_OFF_2 (Object)
ID_VIEW_APPLOOK_OFF_2 (Object)
ID_VIEW_APPLOOK_OFF_X (Object)
ID_VIEW_APPLOOK_VS_20 (Object)
ID_VIEW_APPLOOK_VS_20 (Object)
ID_VIEW_APPLOOK_WIN_ (Object)
ID_VIEW_APPLOOK_WIN_ (Object)
ID_VIEW_APPLOOK_WIND (Object)
ID_VIEW_STATUS_BAR (Object)
ID_VIEW_TOOLBAR (Object)
控件
IDC_COORD (Object)
IDC_EDIT_A (Object)
EN_ALIGN_LTR_EC
EN_ALIGN_RTL_EC
EN_CHANGE
EN_ERRSPACE
EN_HSCROLL
EN_KILLFOCUS
EN_MAXTEXT
EN_SETFOCUS
EN_UPDATE
EN_VSCROLL
NM_THEMECHANGED
<Add> OnEnKillfocusEditA
IDC_EDIT_B (Object)
IDCANCEL (Object)



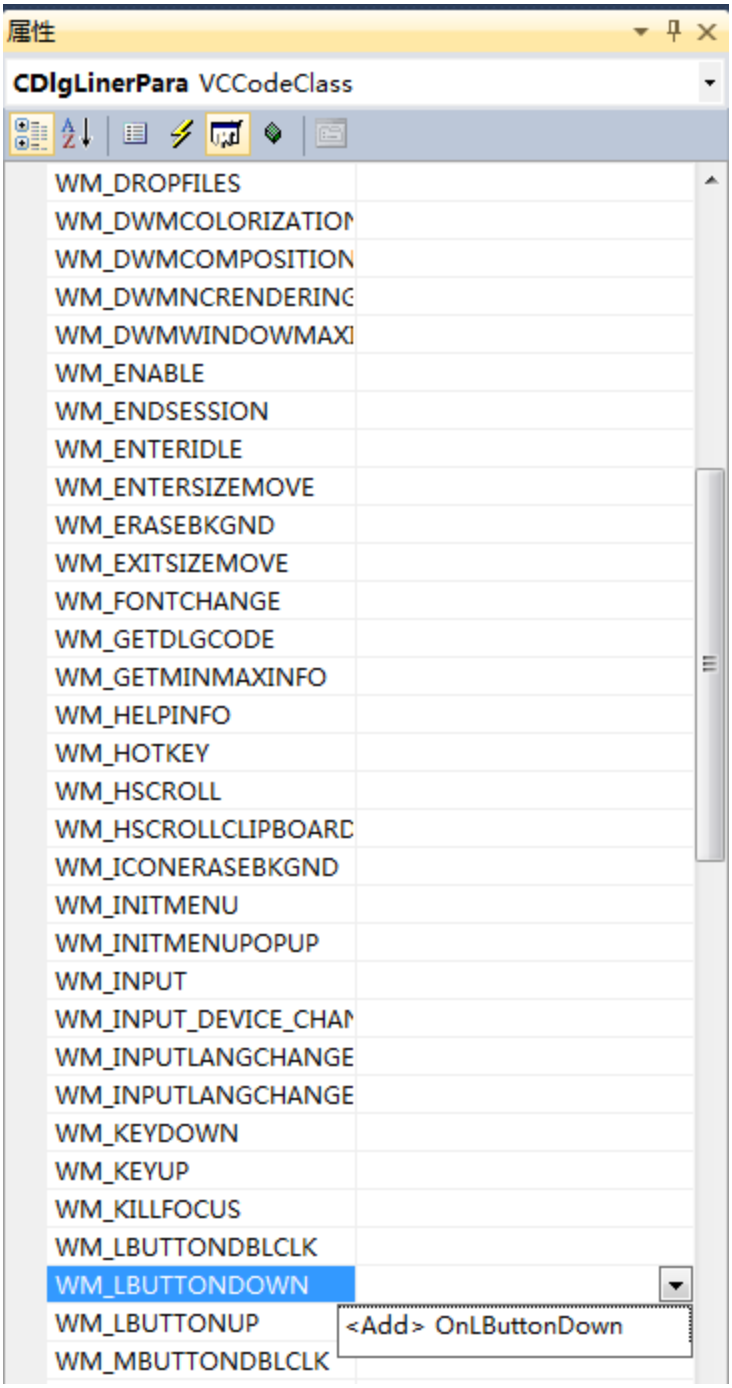
（3）添加消息函数

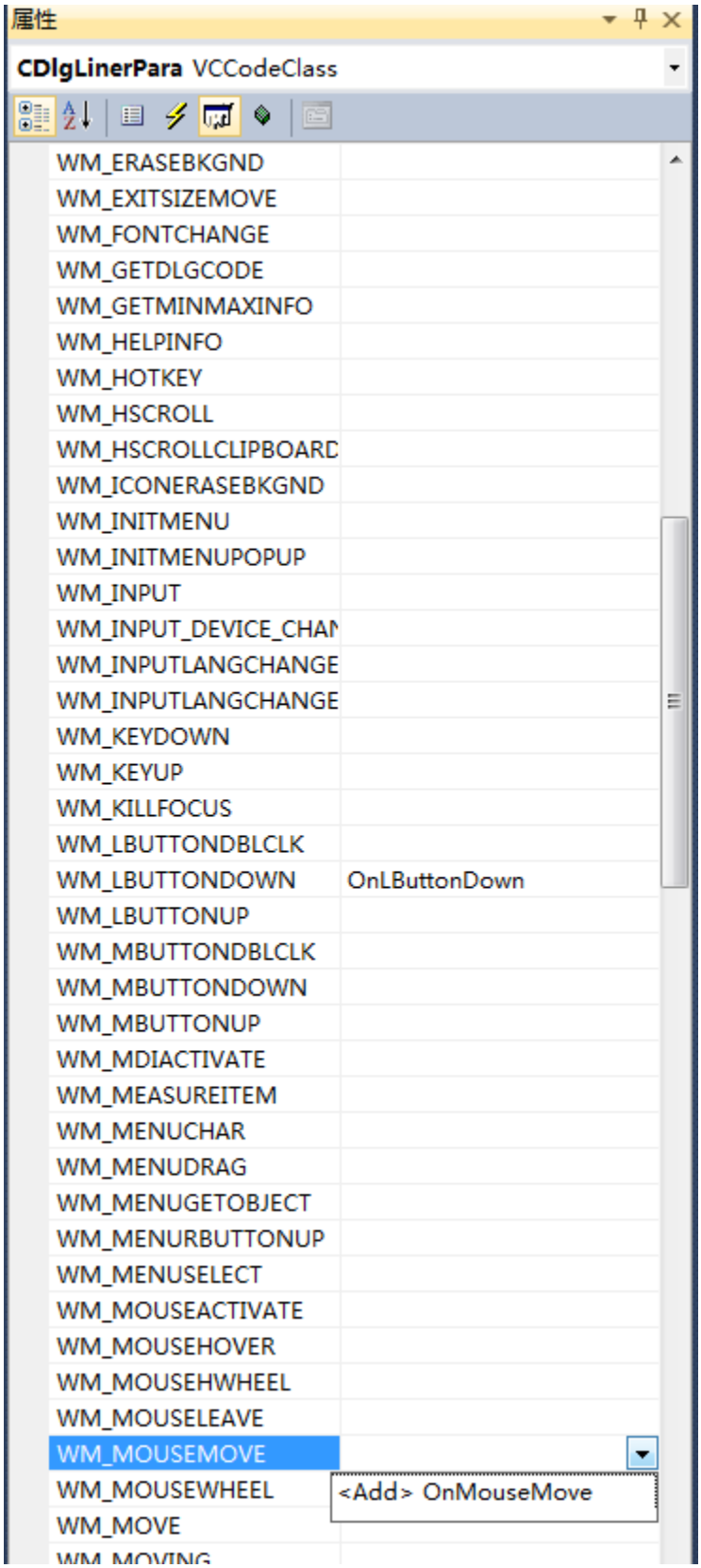
afx_msg void OnLButtonDown(UINT nFlags, CPoint point);

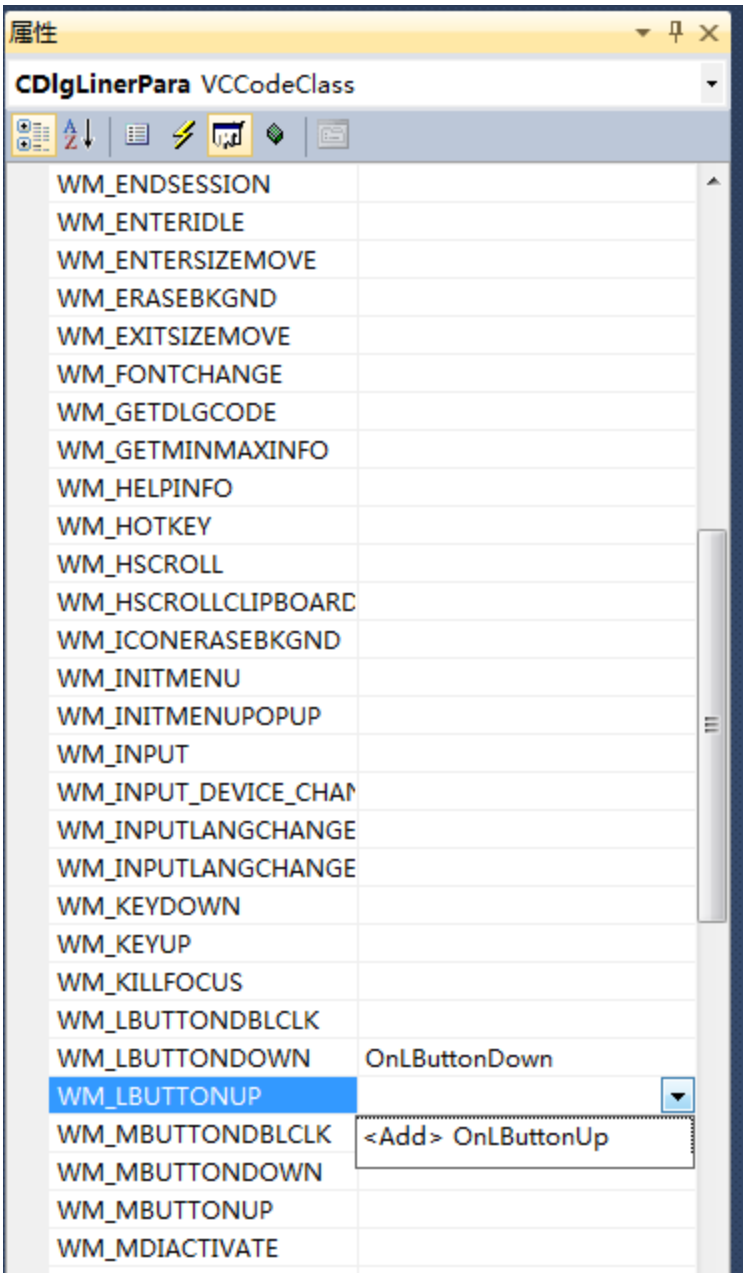
afx_msg void OnMouseMove(UINT nFlags, CPoint point);

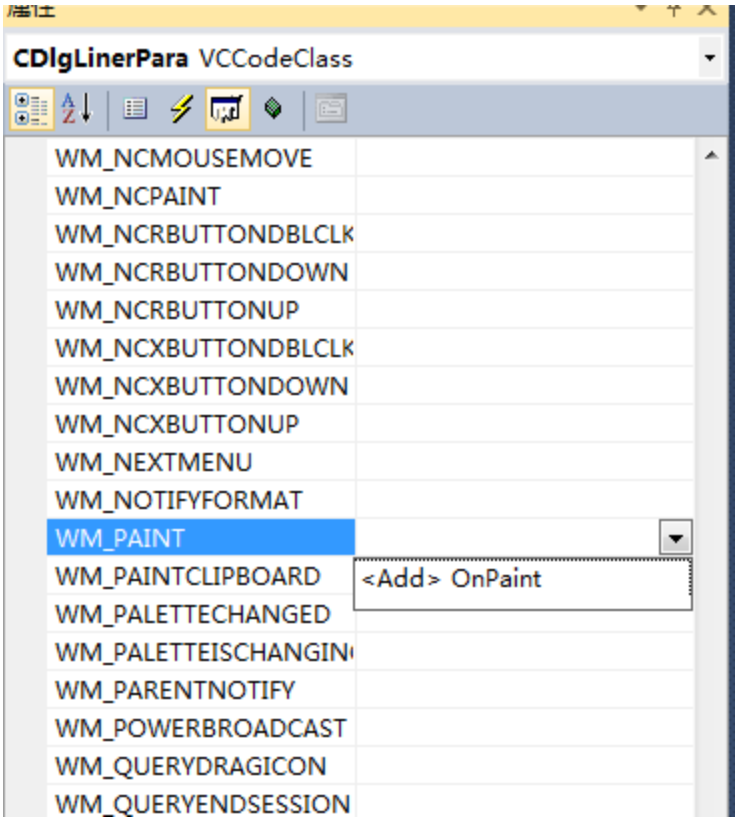
afx_msg void OnLButtonUp(UINT nFlags, CPoint point)

afx_msg void OnPaint();


属性
CDlgLinerPara VCCodeClass
WM_DROPFILES
WM_DWMCOLORIZATION
WM_DWMCOMPOSITION
WM_DWMNCRENDERING
WM_DWMWINDOWMAXI
WM_ENABLE
WM_ENDSESSION
WM_ENTERIDLE
WM_ENTERSIZEMOVE
WM_ERASEBKGND
WM_EXITSIZEMOVE
WM_FONTCHANGE
WM_GETDLGCODE
WM_GETMINMAXINFO
WM_HELPINFO
WM_HOTKEY
WM_HSCROLL
WM_HSCROLLCLIPBOARD
WM_ICONERASEBKGND
WM_INITMENU
WM_INITMENUPOPUP
WM_INPUT
WM_INPUT_DEVICE_CHAN
WM_INPUTLANGCHANGE
WM_INPUTLANGCHANGE
WM_KEYDOWN
WM_KEYUP
WM_KILLFOCUS
WM_LBUTTONDBLCLK
WM_LBUTTONDOWN
WM_LBUTTONUP
<Add> OnLButtonDown
WM_MBUTTONDBLCLK


属性
CDlgLinerPara VCCodeClass
WM_ERASEBKGND
WM_EXITSIZEMOVE
WM_FONTCHANGE
WM_GETDLGCODE
WM_GETMINMAXINFO
WM_HELPINFO
WM_HOTKEY
WM_HSCROLL
WM_HSCROLLCLIPBOARD
WM_ICONERASEBKGND
WM_INITMENU
WM_INITMENUPOPUP
WM_INPUT
WM_INPUT_DEVICE_CHAN
WM_INPUTLANGCHANGE
WM_INPUTLANGCHANGE
WM_KEYDOWN
WM_KEYUP
WM_KILLFOCUS
WM_LBUTTONDBLCLK
WM_LBUTTONDOWN
OnLButtonDown
WM_LBUTTONUP
WM_MBUTTONDBLCLK
WM_MBUTTONDOWN
WM_MBUTTONUP
WM_MDIACTIVATE
WM_MEASUREITEM
WM_MENUCHAR
WM_MENUDRAG
WM_MENUGETOBJECT
WM_MENURBUTTONUP
WM_MENUSELECT
WM_MOUSEACTIVATE
WM_MOUSEHOVER
WM_MOUSEHWHEEL
WM_MOUSELEAVE
WM_MOUSEMOVE
WM_MOUSEWHEEL
<Add> OnMouseMove
WM_MOVE
WM_MOVING


属性
CDlgLinerPara VCCodeClass
WM_ENDSESSION
WM_ENTERIDLE
WM_ENTERSIZEMOVE
WM_ERASEBKGND
WM_EXITSIZEMOVE
WM_FONTCHANGE
WM_GETDLGCODE
WM_GETMINMAXINFO
WM_HELPINFO
WM_HOTKEY
WM_HSCROLL
WM_HSCROLLCLIPBOARD
WM_ICONERASEBKGND
WM_INITMENU
WM_INITMENUPOPUP
WM_INPUT
WM_INPUT_DEVICE_CHAN
WM_INPUTLANGCHANGE
WM_INPUTLANGCHANGE
WM_KEYDOWN
WM_KEYUP
WM_KILLFOCUS
WM_LBUTTONDBLCLK
WM_LBUTTONDOWN
OnLButtonDown
WM_LBUTTONUP
WM_MBUTTONDBLCLK
<Add> OnLButtonUp
WM_MBUTTONDOWN
WM_MBUTTONUP
WM_MDIACTIVATE




```
1.  //（1）   对话框头文件 DlgLinerPara.h
2.  #if !defined(AFX_DLGLINERPARA_H__D969C177_0D07_4F76_9BB8_0D176DDC8E23__INCLU
    DED_)
3.  #define AFX_DLGLINERPARA_H__D969C177_0D07_4F76_9BB8_0D176DDC8E23__INCLUDED_

4.
5.  #if _MSC_VER > 1000
6.  #pragma once
7.  #endif // _MSC_VER > 1000
8.  // DlgLinerPara.h : header file
9.  //
10.
11. ////////////////////////////////////////////////////////////////////////////
    /
12. // CDlgLinerPara dialog
13. class CDlgLinerPara : public CDialog
14. {
15. // Construction
16. public:
17.     // 标识是否已经绘制橡皮筋线
18.     BOOL    m_bDrawed;
19.
20.     // 保存鼠标左键单击时的位置
21.     CPoint  m_p1;
```

```
22.
23.     // 保存鼠标拖动时的位置
24.     CPoint  m_p2;
25.
26.     // 当前鼠标拖动状态，TRUE 表示正在拖动。
27.     BOOL    m_bIsDraging;
28.
29.     // 相应鼠标事件的矩形区域
30.     CRect   m_MouseRect;
31.
32.     CDlgLinerPara(CWnd* pParent = NULL);   // standard constructor
33.
34.
35.
36. // Dialog Data
37.     //{{AFX_DATA(CDlgLinerPara)
38.     enum { IDD = IDD_DLG_LinerPara };
39.     float   m_fA;
40.     float   m_fB;
41.     //}}AFX_DATA
42.
43.
44. // Overrides
45.     // ClassWizard generated virtual function overrides
46.     //{{AFX_VIRTUAL(CDlgLinerPara)
47.     protected:
48.     virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV support
49.     //}}AFX_VIRTUAL
50.
51. // Implementation
52. protected:
53.
54.     // Generated message map functions
55.     //{{AFX_MSG(CDlgLinerPara)
56.     virtual BOOL OnInitDialog();
57.     afx_msg void OnKillfocusEditA();
58.     afx_msg void OnKillfocusEditB();
59.     afx_msg void OnLButtonDown(UINT nFlags, CPoint point);
60.     afx_msg void OnMouseMove(UINT nFlags, CPoint point);
61.     afx_msg void OnLButtonUp(UINT nFlags, CPoint point);
62.     afx_msg void OnPaint();
63.     //}}AFX_MSG
64.     DECLARE_MESSAGE_MAP()
65. };
```

```
66. （2） 对话框代码 DlgLinerPara.cpp
67. // DlgLinerPara.cpp : implementation file
68. //
69.
70. #include "stdafx.h"
71. #include "txsy1.h"
72. #include "DlgLinerPara.h"
73.
74. #ifdef _DEBUG
75. #define new DEBUG_NEW
76. #undef THIS_FILE
77. static char THIS_FILE[] = __FILE__;
78. #endif
79.
80. /////////////////////////////////////////////////////////////////////////
    /
81. // CDlgLinerPara dialog
82.
83.
84. CDlgLinerPara::CDlgLinerPara(CWnd* pParent /*=NULL*/)
85.     : CDialog(CDlgLinerPara::IDD, pParent)
86. {
87.     //{{AFX_DATA_INIT(CDlgLinerPara)
88.     m_fA = 0.0f;
89.     m_fB = 0.0f;
90.     //}}AFX_DATA_INIT
91. }
92.
93.
94. void CDlgLinerPara::DoDataExchange(CDataExchange* pDX)
95. {
96.     CDialog::DoDataExchange(pDX);
97.     //{{AFX_DATA_MAP(CDlgLinerPara)
98.     DDX_Text(pDX, IDC_EDIT_A, m_fA);
99.     DDX_Text(pDX, IDC_EDIT_B, m_fB);
100.     //}}AFX_DATA_MAP
101. }
102.
103.
104. BEGIN_MESSAGE_MAP(CDlgLinerPara, CDialog)
105.     //{{AFX_MSG_MAP(CDlgLinerPara)
106.     ON_EN_KILLFOCUS(IDC_EDIT_A, OnKillfocusEditA)
107.     ON_EN_KILLFOCUS(IDC_EDIT_B, OnKillfocusEditB)
108.     ON_WM_LBUTTONDOWN()
```

```
    ON_WM_MOUSEMOVE()
    ON_WM_LBUTTONUP()
    ON_WM_PAINT()
    //}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
//
// CDlgLinerPara message handlers

BOOL ***::OnInitDialog()
{
    // 调用默认 OnInitDialog 函数
    CDialog::OnInitDialog();
    // 获取绘制参数曲线的标签
    CWnd* pWnd = GetDlgItem(IDC_COORD);
    // 计算接受鼠标事件的有效区域
    pWnd->GetClientRect(m_MouseRect);
    pWnd->ClientToScreen(&m_MouseRect);

    CRect rect;
    GetClientRect(rect);
    ClientToScreen(&rect);

    m_MouseRect.top -= rect.top;
    m_MouseRect.left -= rect.left;

    // 设置接受鼠标事件的有效区域
    m_MouseRect.top += 25;
    m_MouseRect.left += 10;
    m_MouseRect.bottom = m_MouseRect.top + 255;
    m_MouseRect.right = m_MouseRect.left + 256;

    // 初始化拖动状态
    m_bIsDraging = FALSE;

    // 返回 TRUE
    return TRUE;  // return TRUE unless you set the focus to a control
                  // EXCEPTION: OCX Property Pages should return FALSE
}

void ***::OnKillfocusEditA()
{
    // 保存用户设置
```

```
152.     UpdateData(TRUE);
153. 
154.     // 重绘
155.     InvalidateRect(m_MouseRect, TRUE);
156. 
157. }
158. 
159. void ***::OnKillfocusEditB()
160. {
161. // 保存用户设置
162.     UpdateData(TRUE);
163. 
164.     // 重绘
165.     InvalidateRect(m_MouseRect, TRUE);
166. 
167. }
168. 
169. void ***::OnLButtonDown(UINT nFlags, CPoint point)
170. {
171. // 当用户单击鼠标左键开始拖动
172.     if(m_MouseRect.PtInRect(point))
173.     {
174.         // 保存当前鼠标位置
175.         m_p1 = point;
176. 
177.         // 转换坐标系
178.         m_p1.x = m_p1.x - m_MouseRect.left + 10;
179.         m_p1.y = m_p1.y - m_MouseRect.top + 25;
180. 
181.         // 设置拖动状态
182.         m_bIsDraging = TRUE;
183. 
184.         // 设置 m_bDrawed 为 FALSE
185.         m_bDrawed = FALSE;
186. 
187.         // 更改光标
188.         ::SetCursor(::LoadCursor(NULL, IDC_CROSS));
189. 
190.         // 开始跟踪鼠标事件（保证当鼠标移动到窗体外时也可以接收到鼠标释放事件）
191.         SetCapture();
192.     }
193. 
194.     // 默认单击鼠标左键处理事件
195.     //CDialog::OnLButtonDown(nFlags, point);
```

```
196. }
197.
198. void ***::OnMouseMove(UINT nFlags, CPoint point)
199. {
200.     // 判断当前光标是否在绘制区域
201.     if(m_MouseRect.PtInRect(point))
202.     {
203.         // 更改光标
204.         ::SetCursor(::LoadCursor(NULL, IDC_CROSS));
205.
206.         // 判断是否正在拖动
207.         if (m_bIsDraging)
208.         {
209.             // 获取绘图的标签
210.             CWnd* pWnd = GetDlgItem(IDC_COORD);
211.
212.             // 获取设备上下文
213.             CDC* pDC = pWnd->GetDC();
214.
215.             // 设置绘制方式为异或模式
216.             int nOldDrawMode = pDC->SetROP2(R2_XORPEN);
217.
218.             // 创建新的画笔
219.             CPen* pPen = new CPen;
220.             pPen->CreatePen(PS_DOT,1,RGB(0,0,0));
221.
222.             // 选中新画笔
223.             CGdiObject* pOldPen = pDC->SelectObject(pPen);
224.
225.             // 判断是否已经画过橡皮筋线
226.             if (m_bDrawed)
227.             {
228.                 // 擦去以前的橡皮筋线
229.                 pDC->MoveTo(m_p1);
230.                 pDC->LineTo(m_p2);
231.             }
232.
233.             // 保存当前的坐标
234.             m_p2 = point;
235.
236.             // 转换坐标系
237.             m_p2.x = m_p2.x - m_MouseRect.left + 10;
238.             m_p2.y = m_p2.y - m_MouseRect.top + 25;
239.
```

```
240.             // 绘制一条新橡皮筋线
241.             pDC->MoveTo(m_p1);
242.             pDC->LineTo(m_p2);
243.
244.             // 设置 m_bDrawed 为 TRUE
245.             m_bDrawed = TRUE;
246.
247.             // 选回以前的画笔
248.             pDC->SelectObject(pOldPen);
249.
250.             // 恢复成以前的绘制模式
251.             pDC->SetROP2(nOldDrawMode);
252.
253.             delete pPen;
254.             ReleaseDC(pDC);
255.         }
256.     }
257.     else
258.     {
259.         // 判断是否正在拖动
260.         if (m_bIsDraging)
261.         {
262.             // 更改光标
263.             ::SetCursor(::LoadCursor(NULL, IDC_NO));
264.         }
265.     }
266.
267.     // 默认鼠标移动处理事件
268.     //CDialog::OnMouseMove(nFlags, point);
269. }
270.
271. void ***::OnLButtonUp(UINT nFlags, CPoint point)
272. {
273.     // 当用户释放鼠标左键停止拖动
274.     if (m_bIsDraging)
275.     {
276.         // 判断当前光标是否在绘制区域
277.         if(m_MouseRect.PtInRect(point))
278.         {
279.             // 保存当前鼠标位置
280.             m_p2 = point;
281.
282.             // 转换坐标系
283.             m_p2.x = m_p2.x - m_MouseRect.left + 10;
```

```
284.             m_p2.y = m_p2.y - m_MouseRect.top + 25;
285.
286.             if ((m_p1 != m_p2) && (m_p1.x != m_p2.x))
287.             {
288.                 // 转换坐标系
289.                 m_p1.x = m_p1.x - 10;
290.                 m_p1.y = 280 - m_p1.y;
291.                 m_p2.x = m_p2.x - 10;
292.                 m_p2.y = 280 - m_p2.y;
293.
294.                 // 计算斜率和截距
295.                 m_fA = (float) (m_p2.y - m_p1.y) / (m_p2.x - m_p1.x);
296.                 m_fB = m_p1.y - m_fA * m_p1.x;
297.
298.                 // 保存变动
299.                 UpdateData(FALSE);
300.             }
301.
302.             // 重绘
303.             InvalidateRect(m_MouseRect, TRUE);
304.         }
305.         else
306.         {
307.             // 用户在绘制区域外放开鼠标左键
308.
309.             // 获取绘图的标签
310.             CWnd* pWnd = GetDlgItem(IDC_COORD);
311.
312.             // 获取设备上下文
313.             CDC* pDC = pWnd->GetDC();
314.
315.             // 设置绘制方式为异或模式
316.             int nOldDrawMode = pDC->SetROP2(R2_XORPEN);
317.
318.             // 创建新的画笔
319.             CPen* pPen = new CPen;
320.             pPen->CreatePen(PS_DOT,1,RGB(0,0,0));
321.
322.             // 选中新画笔
323.             CGdiObject* pOldPen = pDC->SelectObject(pPen);
324.
325.             // 判断是否已经画过橡皮筋线
326.             if (m_bDrawed)
327.             {
```

```
                // 擦去以前的橡皮筋线
                pDC->MoveTo(m_p1);
                pDC->LineTo(m_p2);
            }

            // 选回以前的画笔
            pDC->SelectObject(pOldPen);

            // 恢复成以前的绘制模式
            pDC->SetROP2(nOldDrawMode);

            delete pPen;
            ReleaseDC(pDC);

        }

        // 解除对鼠标事件的跟踪
        ::ReleaseCapture();

        // 重置拖动状态
        m_bIsDraging = FALSE;
    }

    // 默认释放鼠标左键处理事件
    //CDialog::OnLButtonUp(nFlags, point);

}

void ***::OnPaint()
{
    // 字符串
    CString str;

    // 直线和坐标轴二个交点坐标
    int x1, y1, x2, y2;

    // 设备上下文
    CPaintDC dc(this);

    // 获取绘制坐标的文本框
    CWnd* pWnd = GetDlgItem(IDC_COORD);

    // 指针
    CDC* pDC = pWnd->GetDC();
```

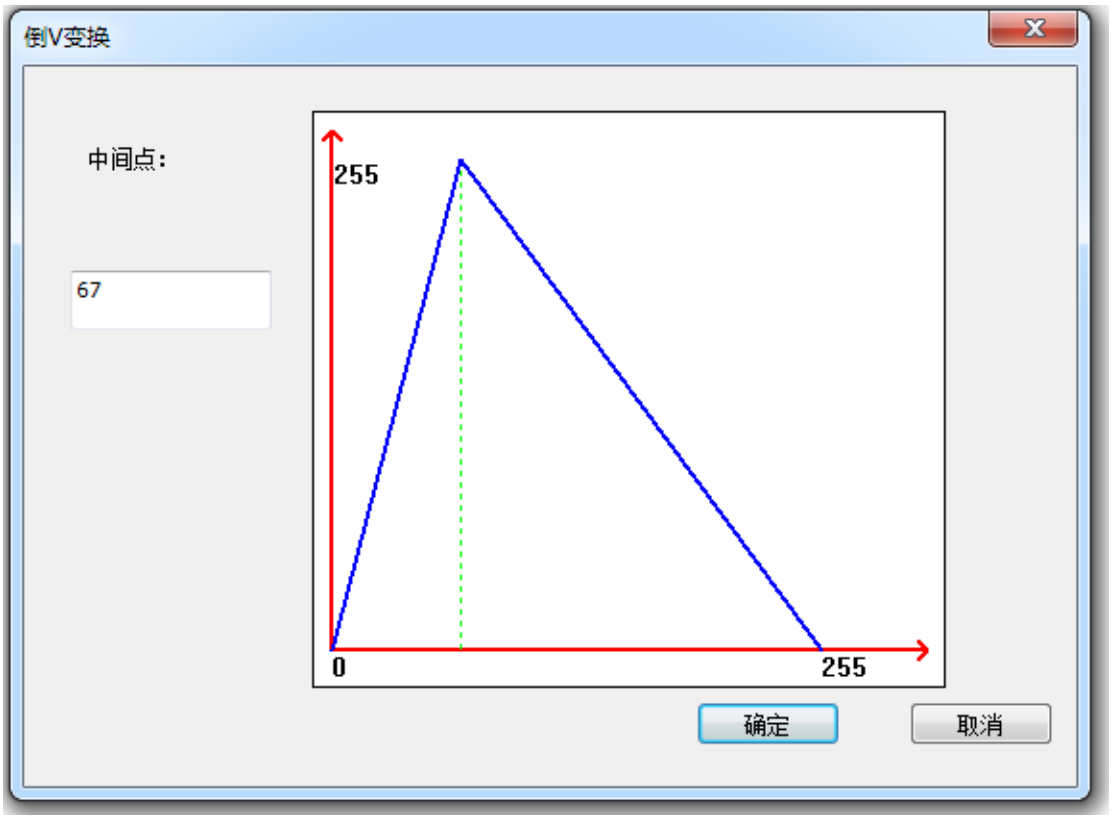
```
372.     pWnd->Invalidate();
373.     pWnd->UpdateWindow();
374.
375.     pDC->Rectangle(0,0,330,300);
376.
377.     // 创建画笔对象
378.     CPen* pPenRed = new CPen;
379.
380.     // 红色画笔
381.     pPenRed->CreatePen(PS_SOLID,2,RGB(255,0,0));
382.
383.     // 创建画笔对象
384.     CPen* pPenBlue = new CPen;
385.
386.     // 蓝色画笔
387.     pPenBlue->CreatePen(PS_SOLID,2,RGB(0,0, 255));
388.
389.     // 选中当前红色画笔，并保存以前的画笔
390.     CGdiObject* pOldPen = pDC->SelectObject(pPenRed);
391.
392.     // 绘制坐标轴
393.     pDC->MoveTo(10,10);
394.
395.     // 垂直轴
396.     pDC->LineTo(10,280);
397.
398.     // 水平轴
399.     pDC->LineTo(320,280);
400.
401.     // 写坐标
402.     str.Format(L"0");
403.     pDC->TextOut(10, 281, str);
404.
405.     str.Format(L"255");
406.     pDC->TextOut(265, 281, str);
407.     pDC->TextOut(11, 25, str);
408.
409.     // 绘制 X 轴箭头
410.     pDC->LineTo(315,275);
411.     pDC->MoveTo(320,280);
412.     pDC->LineTo(315,285);
413.
414.     // 绘制 Y 轴箭头
415.     pDC->MoveTo(10,10);
```

```
    pDC->LineTo(5,15);
    pDC->MoveTo(10,10);
    pDC->LineTo(15,15);

    // 更改成蓝色画笔
    pDC->SelectObject(pPenBlue);

    // 计算直线和坐标轴二个交点坐标
    if (m_fA >= 0)
    {
        if (((m_fA * 255 + m_fB) >= 0) && (m_fB < 255))
        {
            // 计算(x1, y1)坐标
            if (m_fB < 0)
            {
                x1 = (int) (- m_fB/m_fA + 0.5);
                y1 = 0;
            }
            else
            {
                x1 = 0;
                y1 = (int) (m_fB + 0.5);
            }

            // 计算(x2, y2)坐标
            if ((m_fA * 255 + m_fB) > 255)
            {
                x2 = (int) ((255- m_fB)/m_fA + 0.5);
                y2 = 255;
            }
            else
            {
                x2 = 255;
                y2 = (int) (255* m_fA + m_fB + 0.5);
            }
        }
        else if(((m_fA * 255 + m_fB) < 0))
        {
            // 转换后所有象素值都小于 0，直接设置为 0
            x1 = 0;
            y1 = 0;
            x2 = 255;
            y2 = 0;
        }
```

```
460.         else
461.         {
462.             // 转换后所有象素值都大于 255，直接设置为 255
463.             x1 = 0;
464.             y1 = 255;
465.             x2 = 255;
466.             y2 = 255;
467.         }
468.     }
469.     else  // 斜率小于 0
470.     {
471.         if ((m_fB > 0) && (255* m_fA + m_fB < 255))
472.         {
473.             // 计算(x1, y1)坐标
474.             if (m_fB > 255)
475.             {
476.                 x1 = (int) ((255- m_fB)/m_fA + 0.5);
477.                 y1 = 255;
478.             }
479.             else
480.             {
481.                 x1 = 0;
482.                 y1 = (int) (m_fB + 0.5);
483.             }
484.
485.             // 计算(x2, y2)坐标
486.             if ((m_fA * 255 + m_fB) < 0)
487.             {
488.                 x2 = (int) (- m_fB/m_fA + 0.5);
489.                 y2 = 0;
490.             }
491.             else
492.             {
493.                 x2 = 255;
494.                 y2 = (int) (255* m_fA + m_fB + 0.5);
495.             }
496.         }
497.         else if (m_fB <=0)
498.         {
499.             // 转换后所有象素值都小于 0，直接设置为 0
500.             x1 = 0;
501.             y1 = 0;
502.             x2 = 255;
503.             y2 = 0;
```

```
        }
        else
        {
            // 转换后所有象素值都大于 255，直接设置为 255
            x1 = 0;
            y1 = 255;
            x2 = 255;
            y2 = 255;
        }
    }

    // 绘制坐标值
    str.Format(L"(%d, %d)", x1, y1);
    pDC->TextOut(x1 + 10, 280 - y1 + 1, str);
    str.Format(L"(%d, %d)", x2, y2);
    pDC->TextOut(x2 + 10, 280 - y2 + 1, str);

    // 绘制用户指定的线性变换直线（注意转换坐标系）
    pDC->MoveTo(x1 + 10, 280 - y1);
    pDC->LineTo(x2 + 10, 280 - y2);

    // 恢复以前的画笔
    pDC->SelectObject(pOldPen);

    // 绘制边缘
    pDC->MoveTo(10,25);
    pDC->LineTo(265,25);
    pDC->LineTo(265,280);

    // 删除新的画笔
    delete pPenRed;
    delete pPenBlue;
}
```

编译执行。



四、实验心得

通过本次实验,我对 MFC 框架有了进一步的认识，并结合数字图像处理专业课程的内容,对图片进行线性变换，还能保存显示图片，增加了新的 C++处理数字图像的技巧，收获良多,对于以后使用机器学习处理数字图像奠定下良好的理论基础。