实验一 直线绘制

1. 直线绘制：建立 MFC 单文档项目
   (1) 采用 MFC 的绘图函数来绘制直线
   (2) 采用直线绘制算法来绘制直线

【实验过程及编码】

（一）实验步骤

（1）建立 MFC 单文档项目
（2）设计直线类 CLineColor
（3）交互式绘制直线

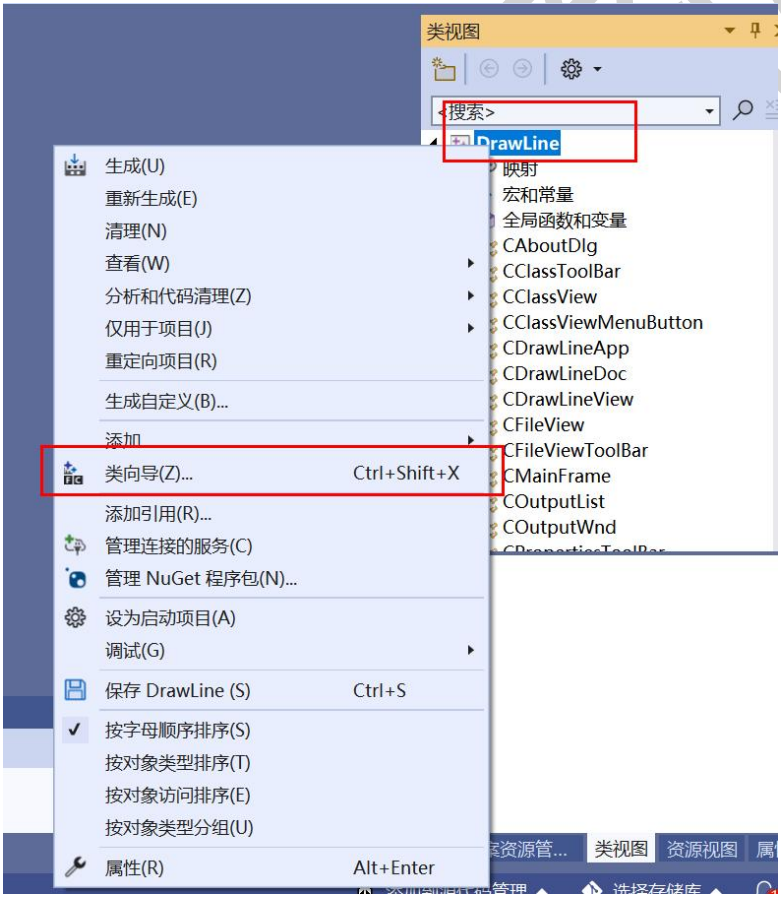
（二）实验编码

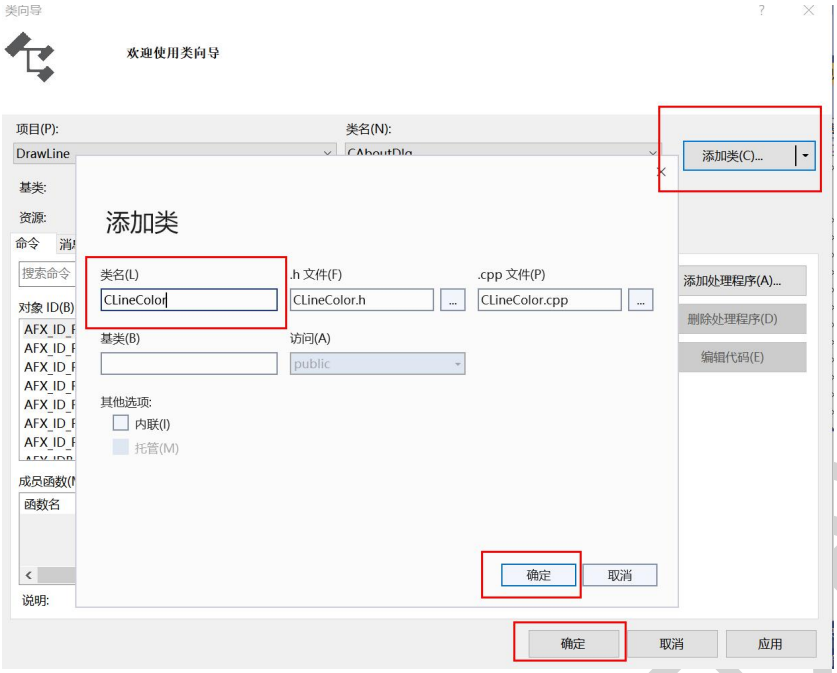
（1）建立 MFC 单文档项目

➢ 新建 MFC 单文档项目 DrawLine

（2）设计直线类 CLineColor

➢ 打开“类向导……”





➢ CLineColor.h

```
class CLineColor
{
private:
    CPoint P0, P1;
public:
    CLineColor();
    CLineColor(CPoint p0, CPoint p1);
    void SetStart(CPoint p0);
    void SetEnd(CPoint p1);
    CPoint GetStart();
    CPoint GetEnd();
    ~CLineColor();
    void MFCLine(CDC* pDC);
    void BresenhamLine(CDC* pDC);
};
```

➢ CLineColor.cpp

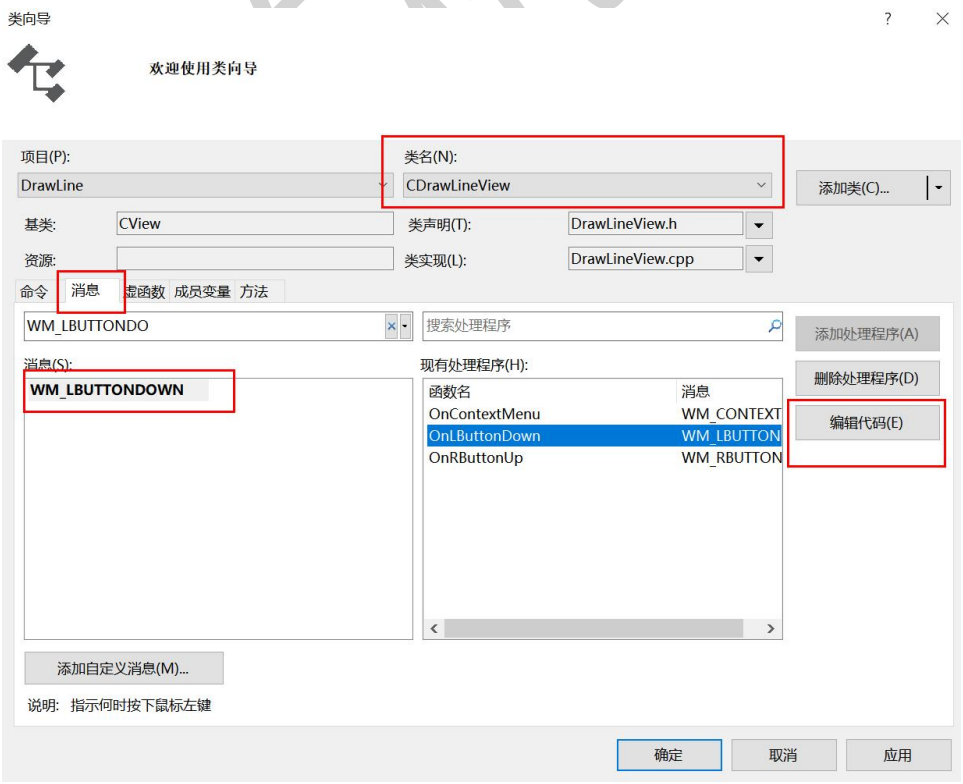
```
#include "CLineColor.h"
CLineColor::CLineColor()
{
    P0 = CPoint(100, 100);
    P1 = CPoint(200, 200);
}
CLineColor::CLineColor(CPoint p0, CPoint p1)
{
    P0 = p0;
    P1 = p1;
}
```

```
CLineColor::~CLineColor()
{
}

void CLineColor::SetStart(CPoint p0)
{
    P0 = p0;
}

void CLineColor::SetEnd(CPoint p1)
{
    P1 = p1;
}

CPoint CLineColor::GetEnd()
{
    return P1;
}
CPoint CLineColor::GetStart()
{
    return P0;
}

void CLineColor::MFCLine(CDC* pDC)   //采用 MFC 中的直线绘制函数来绘制直线
{
    pDC->MoveTo(P0);
    pDC->LineTo(P1);
}

void CLineColor::BresenhamLine(CDC* pDC)//采用直线绘制算法来绘制直线
{
    int dx = abs(P1.x - P0.x);
    int dy = abs(P1.y - P0.y);
    bool bInterChange = TRUE; //主位移方向为 x 轴
    int e, signX, signY, temp;
    //像素是否变化一个单位，+1、-1 或 0
    signX = (P1.x > P0.x) ? 1 : ((P1.x < P0.x) ? -1 : 0);
    signY = (P1.y > P0.y) ? 1 : ((P1.y < P0.y) ? -1 : 0);
    if (dy > dx) //主位移方向更新为 x 轴，实际为 y 轴
    {
        temp = dx;
        dx = dy;
        dy = temp;
        bInterChange = FALSE;
```

```
    }
    e = -dx; //e 的初始值
    CPoint p = P0; //从起点开始
    for (int i = 1; i <= dx; i++)
    {
        pDC->SetPixelV(p.x, p.y, RGB(0,0,255));//画点
        if (bInterChange)//x 轴主位移方向，对 x 坐标进行操作
            p.x += signX;
        else    //y 轴主位移方向，对 y 坐标进行操作
            p.y += signY;
        e += 2 * dy; //递推误差项
        if (e >= 0) //更新，两个方向均执行+1
        {
            if (bInterChange) //x 轴主位移方向，对 y 坐标进行操作
                p.y += signY;
            else //y 轴主位移方向，对 x 坐标进行操作
                p.x += signX;
            e -= 2 * dx; //递推项更新
        }
    }
}]PTemp;
}
```

（3）交互式绘制直线

- 鼠标左键按下选择起点和终点，打开“类向导”添加消息 WM_LBUTTONDOW，并双击命令添加处理函数



➢ 在鼠标左键按下响应函数中编辑代码，调用直线绘制函数

```
CLineColor line;
void CDrawLineView::OnLButtonDown(UINT nFlags, CPoint point)
{
    // TODO: 在此添加消息处理程序代码和/或调用默认值
    if (k)  //在类中定义布尔型数据成员 k，即：BOOL k=true;
        line.SetStart(point);
    else
    {
        line.SetEnd(point);
        CDC *pDC = GetDC();
        //line.MFCLine(pDC);
        line.BresenhamLine(pDC);
        ReleaseDC(pDC);
    }
    k = !k;
    CView::OnLButtonDown(nFlags, point);
}
```