模块 2-2 裁剪算法

一 实验目的

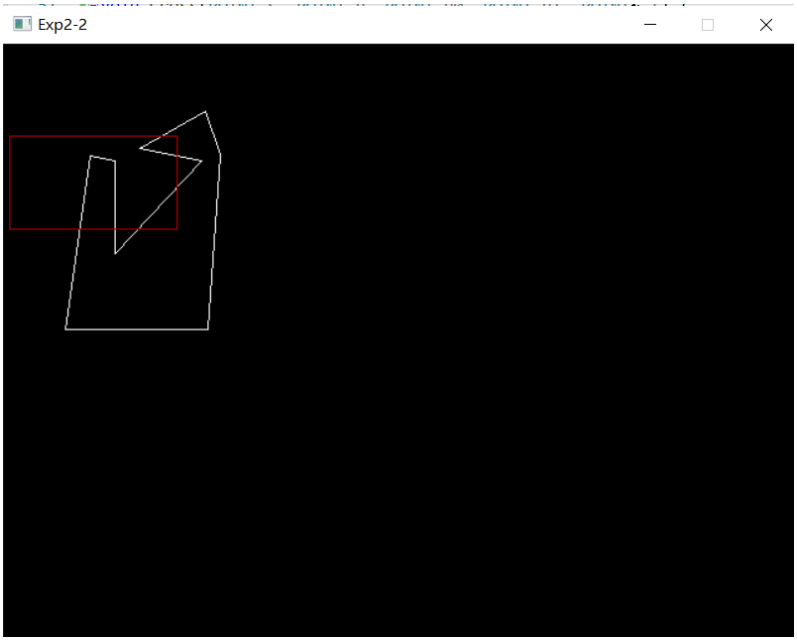
1. 编写直线段、多边形裁剪算法
2. 熟悉逐边裁剪法、Weiler-Atherton 裁剪法的使用

二 实验内容

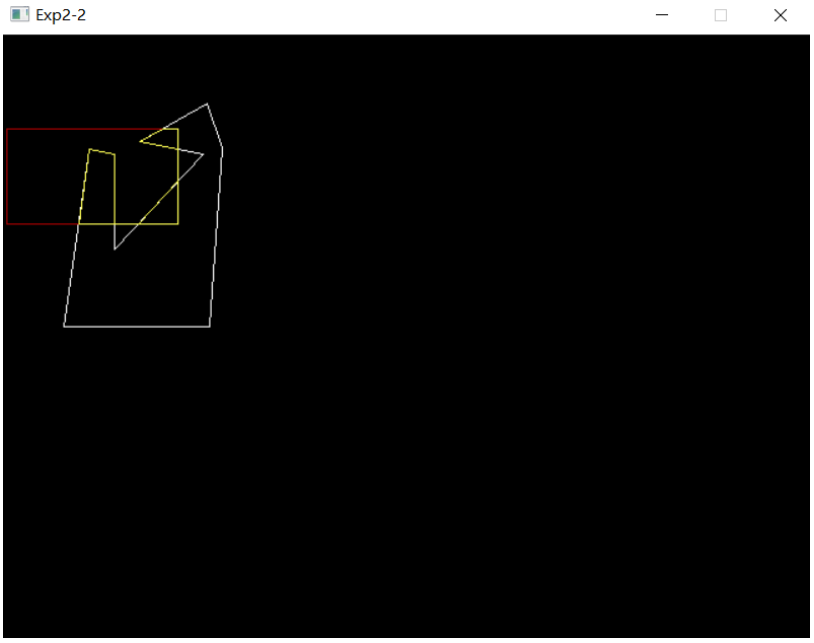
4：用逐边裁剪法实现多边形裁剪（代码最上方功能区注明是否处理退化边）

无退化实验结果如下图所示：

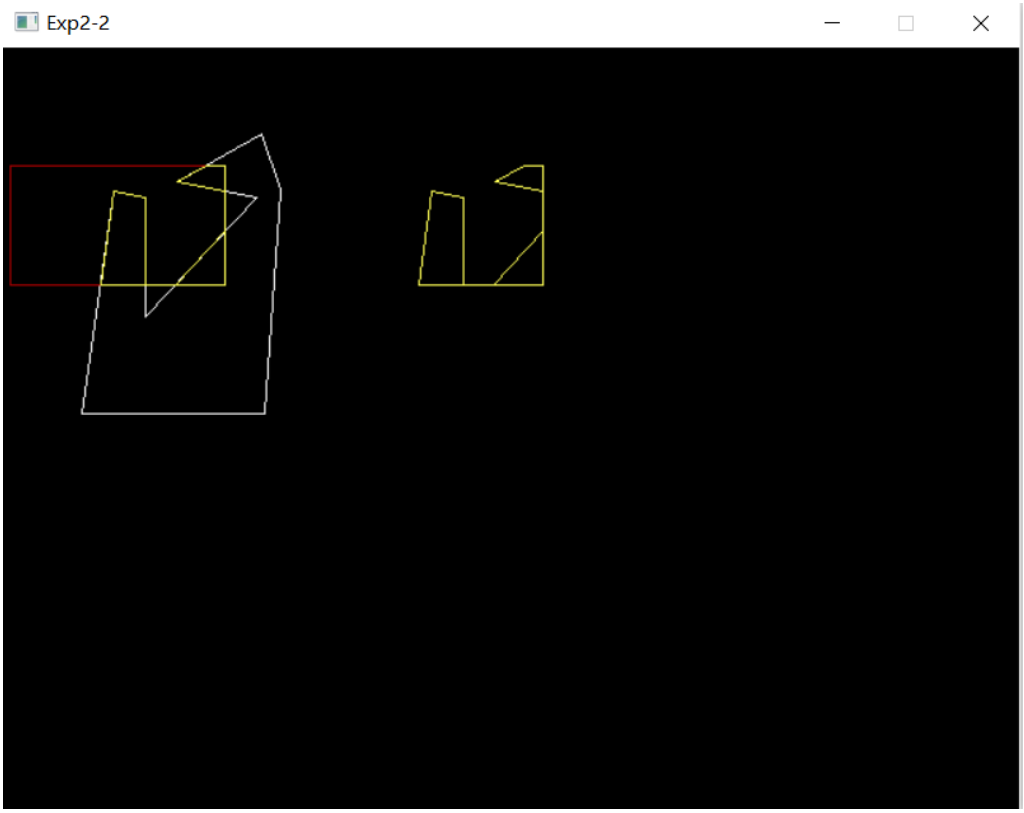
图形初始化：（红色部分为裁剪框，白色部分为待裁剪多边形）



点击左键进行裁剪：（黄色部分为裁剪后剩余多边形）

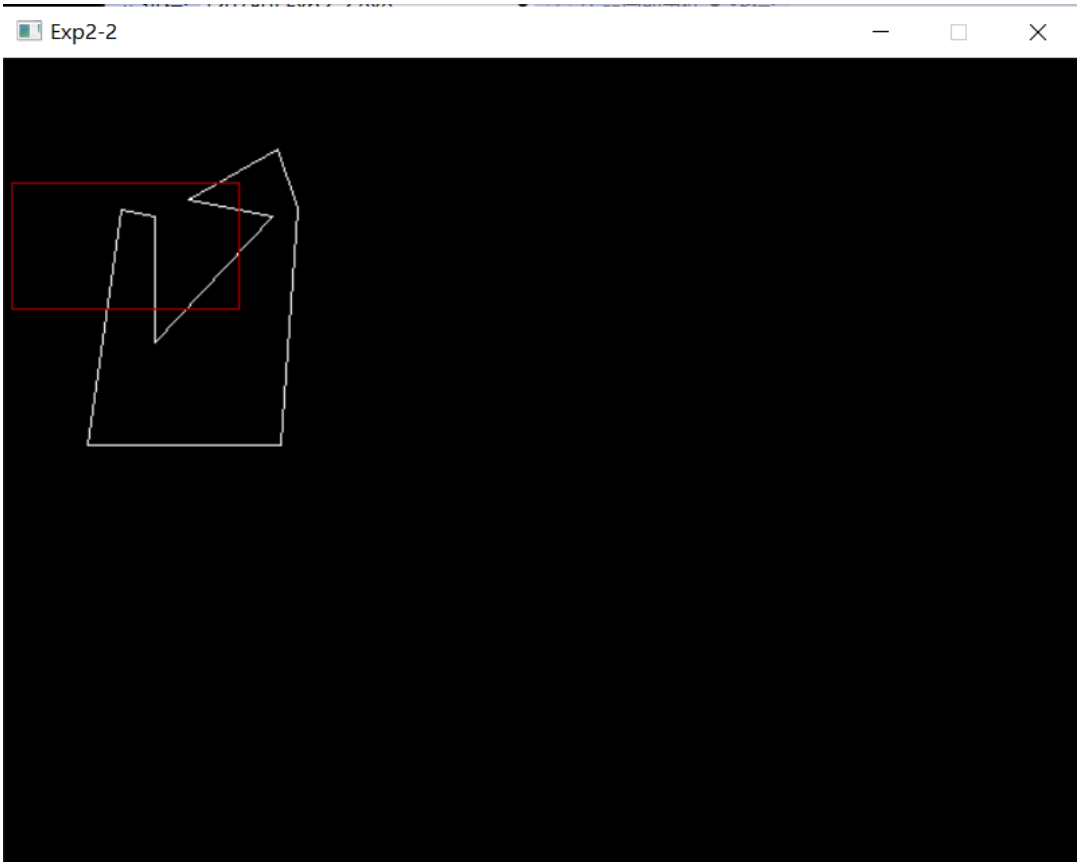


点击右键将裁剪图案右移：（黄色部分为裁剪后剩余多边形）

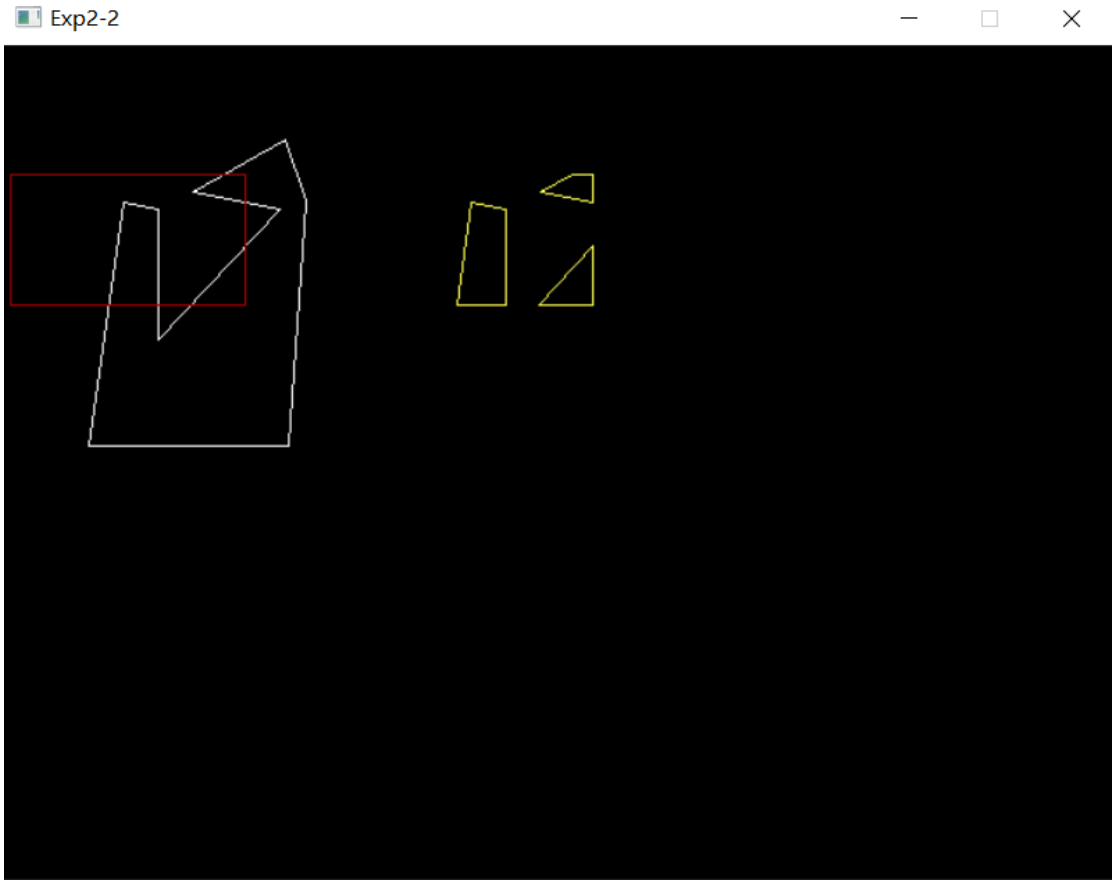


有退化实验结果如下图所示：

图形初始化：（红色部分为裁剪框，白色部分为待裁剪多边形）



点击左键将裁剪图案右移：（黄色部分为裁剪后剩余多边形）



5：用 Weiler-Atherton 裁剪法实现多边形裁剪

实验结果如下图所示：

说明：白色部分为原始多边形，红色部分为裁剪框，黄色部分为通过 Weiler-Atherton 裁剪法实现的裁剪多边形。

总结：

（1） 上述两种裁剪法的代码实现方法，大致是按照 PPT 上的算法思路构建的。原始多边形是在全局里面预设的，由于时间紧张没有测试过其他多边形（以及水平很菜），可能存在 bug 没有找到并维护。望谅解。

（2） 在实现本实验的过程中，调用了之前实验的函数（CSLineClip 等），算是一种学以致用吧。

三 程序说明

Project 中程序的调用：

将当前 cpp 文件的属性——常规——从生成中排除中选择否，其他文件选择是，即可运行当前的 cpp 文件

4 题：无退化

```
//////////////////////////////////////////////////////
// 程序名称：多边形裁剪1-1
// 功    能：用逐边裁剪法实现多边形裁剪（无退化）
// 编译环境：VS2019，EasyX_20220116
// 作    者：夏婉可<2020301010225><1597493790@qq.com>
// 最后修改：2022-3-31

#include <graphics.h>
#include <conio.h>
#include <iostream>
using namespace std;

//框的边界
float XL = 5, XR = 140, YB = 74, YT = 149;
POINT Edge[] = { {XR,YB},{XR,YT},{XL,YT},{XL,YB} };
//自定义多边形
POINT Vertex[] = { {110,84},{160,94},{90,169},{90,94},{70,90},{50,230},
{165,230},{175,89},{163,54} };
int inlen = 9;
#define max 100

//判断顶点和裁剪边的内外关系
bool Inside(POINT test, POINT p0, POINT p1) {
    if (p1.x > p0.x) {
        //裁剪边是窗口的下边
        if (test.y >= p0.y) {
            return 1;
        }
    }
    else if (p1.x < p0.x) {
        //裁剪边是窗口的上边
        if (test.y <= p0.y) {
```

```
            return 1;
        }
    }
    else if (p1.y > p0.y) {
        //裁剪边是窗口的右边
        if (test.x <= p0.x) {
            return 1;
        }
    }
    else if (p1.y < p0.y) {
        //裁剪边是窗口的左边
        if (test.x >= p0.x) {
            return 1;
        }
    }
    return 0;
}

//求多边形的一条边和裁剪边的交点
void Cross(POINT s, POINT p, POINT p0, POINT p1, POINT &i) {
    if (p0.y == p1.y) {
        //水平裁剪边
        i.y = p0.y;
        i.x = s.x + (p0.y - s.y) * (p.x - s.x) / (p.y - s.y);
    }
    else {
        //竖直裁剪边
        i.x = p0.x;
        i.y = s.y + (p0.x - s.x) * (p.y - s.y) / (p.x - s.x);
    }
}

//将新的多边形顶点加入原有顶点组
void Insert(POINT newpoint, int &mylen, POINT p[]) {
    p[mylen].x = newpoint.x;
    p[mylen].y = newpoint.y;
    mylen++;
    //顶点数+=1
}

//裁剪算法
void SHClip(int mylen, POINT in[], int& outlen, POINT out[], POINT p0, POINT p1) {
    POINT s, p, i;
    outlen = 0;
    s = in[mylen - 1];
    for (int j = 0; j < mylen; j++) {
        p = in[j];
        if (Inside(p, p0, p1)) {
            if (Inside(s, p0, p1)) {
                Insert(p, outlen, out);
            }
            else {
                Cross(s, p, p0, p1, i);
                Insert(i, outlen, out);
                Insert(p, outlen, out);
            }
```

```
            }
            else if (Inside(s, p0, p1)) {
                Cross(s, p, p0, p1, i);
                Insert(i, outlen, out);
            }
            s = p;
        }
    }

    int main() {
        //接收框的信息
        float x0, y0, x1, y1;

        initgraph(640, 480);

        //绘制自定的point多边形
        setcolor(WHITE);
        polygon(Vertex, 9);

        //绘制框
        setlinecolor(RED);
        line(XL, YT, XR, YT);
        line(XL, YB, XR, YB);
        line(XL, YT, XL, YB);
        line(XR, YT, XR, YB);

        ExMessage m;
        POINT outp1[max], outp2[max], outp3[max], outp4[max];
        int len1, len2, len3, len4;
        int times = 0;

        while (1) {
            m = getmessage(EX_MOUSE | EX_KEY);
            //用户点击左键后生成裁剪图形
            if (m.message == WM_LBUTTONDOWN) {

                //裁剪过程
                //POINT Edge[] = { {XR,YB},{XR,YT},{XL,YT},{XL,YB} };
                //右边窗口裁剪边
                SHClip(inlen, Vertex, len1, outp1, Edge[0], Edge[1]);
                //上边窗口裁剪边
                SHClip(len1, outp1, len2, outp2, Edge[1], Edge[2]);
                //左边窗口裁剪边
                SHClip(len2, outp2, len3, outp3, Edge[2], Edge[3]);
                //下边窗口裁剪边
                SHClip(len3, outp3, len4, outp4, Edge[3], Edge[0]);

                //连线过程
                setcolor(YELLOW);
                polygon(outp4, len4);
                //原来的位置进行黄色标注裁剪

                times++;
            }
            //用户点击右键后在旁边空白处生成裁剪图形
            else if (m.message == WM_RBUTTONDOWN) {
```

```
            for (int i = 0; i < len4; i++) {
                outp4[i].x += 200;
            }
            setcolor(YELLOW);
            polygon(outp4, len4);
            times++;
        }
        if (times == 2) {
            break;
        }
    }

    _getch();                   // 按任意键继续
    closegraph();           // 关闭绘图窗口
    return 0;
}
```

4题：有退化

```
//////////////////////////////////////////////////////
// 程序名称：多边形裁剪1-2
// 功    能：用逐边裁剪法实现多边形裁剪（有退化）
// 编译环境：VS2019，EasyX_20220116
// 作    者：夏婉可<2020301010225><1597493790@qq.com>
// 最后修改：2022-3-31

#include <graphics.h>
#include <conio.h>
#include <iostream>
#include <stdlib.h>
#include <stdio.h>
using namespace std;

//框的边界
float XL = 5, XR = 140, YB = 74, YT = 149;
POINT Edge[] = { {XR,YB},{XR,YT},{XL,YT},{XL,YB} };
//自定义多边形
POINT Vertex[] = { {110,84},{160,94},{90,169},{90,94},{70,90},{50,230},
{165,230},{175,89},{163,54} };
int inlen = 9;
#define max 100

//判断顶点和裁剪边的内外关系
bool Inside(POINT test, POINT p0, POINT p1) {
    if (p1.x > p0.x) {
        //裁剪边是窗口的下边
        if (test.y >= p0.y) {
            return 1;
        }
    }
    else if (p1.x < p0.x) {
        //裁剪边是窗口的上边
        if (test.y <= p0.y) {
            return 1;
        }
    }
    else if (p1.y > p0.y) {
```

```
        //裁剪边是窗口的右边
        if (test.x <= p0.x) {
            return 1;
        }
    }
    else if (p1.y < p0.y) {
        //裁剪边是窗口的左边
        if (test.x >= p0.x) {
            return 1;
        }
    }
    return 0;
}

//求多边形的一条边和裁剪边的交点
void Cross(POINT s, POINT p, POINT p0, POINT p1, POINT& i) {
    if (p0.y == p1.y) {
        //水平裁剪边
        i.y = p0.y;
        i.x = s.x + (p0.y - s.y) * (p.x - s.x) / (p.y - s.y);
    }
    else {
        //竖直裁剪边
        i.x = p0.x;
        i.y = s.y + (p0.x - s.x) * (p.y - s.y) / (p.x - s.x);
    }
}

//将新的多边形顶点加入原有顶点组
void Insert(POINT newpoint, int& mylen, POINT p[]) {
    p[mylen].x = newpoint.x;
    p[mylen].y = newpoint.y;
    mylen++;
    //顶点数+=1
}

//裁剪算法
void SHClip(int mylen, POINT in[], int& outlen, POINT out[], POINT p0, POINT p1) {
    POINT s, p, i;
    outlen = 0;
    s = in[mylen - 1];
    for (int j = 0; j < mylen; j++) {
        p = in[j];
        if (Inside(p, p0, p1)) {
            if (Inside(s, p0, p1)) {
                Insert(p, outlen, out);
            }
            else {
                Cross(s, p, p0, p1, i);
                Insert(i, outlen, out);
                Insert(p, outlen, out);
            }
        }
        else if (Inside(s, p0, p1)) {
            Cross(s, p, p0, p1, i);
            Insert(i, outlen, out);
```

```
            }
            s = p;
        }
    }

    //VS不让用快排啊=。=
    int cmp(void* a, void* b) {
        return *(int*)a - *(int*)b;
    }

    int main() {
        //接收框的信息
        initgraph(640, 480);

        //绘制自定的point多边形
        setcolor(WHITE);
        polygon(Vertex, 9);

        //绘制框
        setlinecolor(RED);
        line(XL, YT, XR, YT);
        line(XL, YB, XR, YB);
        line(XL, YT, XL, YB);
        line(XR, YT, XR, YB);

        ExMessage m;
        POINT outp1[max], outp2[max], outp3[max], outp4[max];
        int len1, len2, len3, len4;
        int times = 0;

        while (1) {
            m = getmessage(EX_MOUSE | EX_KEY);
            //用户点击左键后生成裁剪图形
            if (m.message == WM_LBUTTONDOWN) {
                //裁剪过程
                //POINT Edge[] = { {XR,YB},{XR,YT},{XL,YT},{XL,YB} };
                //右边窗口裁剪边
                SHClip(inlen, Vertex, len1, outp1, Edge[0], Edge[1]);
                //上边窗口裁剪边
                SHClip(len1, outp1, len2, outp2, Edge[1], Edge[2]);
                //左边窗口裁剪边
                SHClip(len2, outp2, len3, outp3, Edge[2], Edge[3]);
                //下边窗口裁剪边
                SHClip(len3, outp3, len4, outp4, Edge[3], Edge[0]);

                //连线过程
                //setcolor(YELLOW);
                //polygon(outp4, len4);
                //原来的位置进行黄色标注裁剪

                /*closegraph();            // 关闭绘图窗口
                for (int t = 0; t < len4; t++) {
                    cout << outp4[t].x <<" " << outp4[t].y << endl;

                }*/
```

```
            //退化边，根据outp4修改吧
            for (int i = 0; i < len4 - 1; i++) {
                int flag = 1;
                if (outp4[i].x == outp4[i + 1].x) {
                    if (outp4[i].x == int(XL) || outp4[i].x == int(XR)) {
                        flag = 0;
                    }
                }
                if (outp4[i].y == outp4[i + 1].y) {
                    if (outp4[i].y == int(YB) || outp4[i].y == int(YT)) {
                        flag = 0;
                    }
                }
                if (flag == 1) {
                    setcolor(YELLOW);
                    //+200像素，可能是为了，方便展示orz
                    line(outp4[i].x + 200, outp4[i].y, outp4[i + 1].x + 200,
outp4[i + 1].y);
                }
            }

            //float XL = 5, XR = 140, YB = 74, YT = 149;
            //{XR,YB},{XR,YT},{XL,YT},{XL,YB}
            //{ {110,84},{160,94},{90,169},{90,94},{70,90},{50,230},
{165,230},{175,89},{163,54} };

            //将坐标值按从小到大排序，奇数线段依次连接。
            int xl[10], xr[10], yb[10], yt[10];//记录和框重合的点
            int cntl = 0, cntr = 0, cntb = 0, cntt = 0;
            for (int t = 0; t < len4; t++) {
                if (outp4[t].x == int(XL)) {
                    //有了一个x坐标了我还用啥二维数组存坐标啊，笑
                    //[XL,y]
                    xl[cntl++] = outp4[t].y;
                }
                if (outp4[t].x == int(XR)) {
                    //[XR,y]
                    xr[cntr++] = outp4[t].y;
                }
                if (outp4[t].y == int(YB)) {
                    //[x,YB]
                    yb[cntb++] = outp4[t].x;
                }
                if (outp4[t].y == int(YT)) {
                    //[x,YB]
                    yt[cntt++] = outp4[t].x;
                }
            }

            /*void selectSort(int& a, int alen) {
                for (int i = 0; i < alen - 1; i++) {
                    int min = i;
                    for (int j = i + 1; j < alen; j++) {
                        if (a[j] < a[min]) {
                            min = j;
                        }
```

```
                }
                //swap elements
                int temp = a[min];
                a[min] = a[i];
                a[i] = temp;
            }
        }*/

        //为什么调用sort函数呢，突然的bug猝不及防。还是传统方法改好了。

        //排序x,y坐标
        //画left边
        if (cntl != 0) {
            for (int i = 0; i < cntl - 1; i++) {
                int min = i;
                for (int j = i + 1; j < cntl; j++) {
                    if (xl[j] < xl[min]) {
                        min = j;
                    }
                }
                int t = xl[min];
                xl[min] = xl[i];
                xl[i] = t;
            }
            for (int i = 0; i < cntl; i += 2) {
                setcolor(YELLOW);
                line(int(XL) + 200, xr[i], int(XL) + 200, xr[i + 1]);
            }
        }
        //画right边
        if (cntr != 0) {
            for (int i = 0; i < cntr - 1; i++) {
                int min = i;
                for (int j = i + 1; j < cntr; j++) {
                    if (xr[j] < xr[min]) {
                        min = j;
                    }
                }
                int t = xr[min];
                xr[min] = xr[i];
                xr[i] = t;
            }
            for (int i = 0; i < cntr; i += 2) {
                setcolor(YELLOW);
                line(int(XR) + 200, xr[i], int(XR) + 200, xr[i + 1]);
            }
        }
        //画bottom边
        if (cntb != 0) {
            for (int i = 0; i < cntb - 1; i++) {
                int min = i;
                for (int j = i + 1; j < cntb; j++) {
                    if (yb[j] < yb[min]) {
                        min = j;
                    }
                }
                }
```

```
                    int t = yb[min];
                    yb[min] = yb[i];
                    yb[i] = t;
                }
                for (int i = 0; i < cntb; i += 2) {
                    setcolor(YELLOW);
                    line(yb[i] + 200, int(YB), yb[i + 1] + 200, int(YB));
                }
            }
            //画top边
            if (cntt != 0) {
                for (int i = 0; i < cntt - 1; i++) {
                    int min = i;
                    for (int j = i + 1; j < cntt; j++) {
                        if (yt[j] < yt[min]) {
                            min = j;
                        }
                    }
                    int t = yt[min];
                    yt[min] = yt[i];
                    yt[i] = t;
                }
                for (int i = 0; i < cntt; i += 2) {
                    setcolor(YELLOW);
                    line(yt[i] + 200, int(YT), yt[i + 1] + 200, int(YT));
                }
            }
        }
    }

    _getch();                   // 按任意键继续
    closegraph();           // 关闭绘图窗口
    return 0;
}
```

5 题

```
////////////////////////////////////////////////////
// 程序名称：多边形裁剪2
// 功    能：用Weiler-Atherton裁剪法实现多边形裁剪
// 编译环境：VS2019，EasyX_20220116
// 作    者：夏婉可<2020301010225><1597493790@qq.com>
// 最后修改：2022-3-31

#include <graphics.h>
#include <conio.h>
#include <iostream>
using namespace std;

//框的边界
float XL = 20, XR = 150, YB = 200, YT = 450;
POINT Edge[] = { {XR,YB},{XR,YT},{XL,YT},{XL,YB} };
//自定义多边形，顺时针排序点坐标
POINT Vertex[] = { {300,100},{100,400},{200,120},{50,300},{100,100} };
int inlen = 5, outlen = 0;
int keepx[100], keepy[100];
```

```
//编码数值
#define LEFT 1
#define RIGHT 2
#define BOTTOM 4
#define TOP 8

//编码函数
int encode(float x, float y, int* code) {
    int c = 0;
    if (x < XL) {
        c = c | LEFT;
    }
    else if (x > XR) {
        c = c | RIGHT;
    }
    if (y < YB) {
        c = c | BOTTOM;;
    }
    else if (y > YT) {
        c = c | TOP;
    }
    *code = c;
    return 0;
}

//CS裁剪
int CSLineClip(float x1, float y1, float x2, float y2) {
    //记录原始点
    float x10 = x1, y10 = y1, x20 = x2, y20 = y2;

    int code1, code2, code;
    float x, y;
    encode(x1, y1, &code1);
    encode(x2, y2, &code2);
    while (code1 != 0 || code2 != 0) {
        if ((code1 & code2) != 0) {
            return 0;
        }
        code = code1;
        if (code1 == 0) {
            code = code2;
        }
        //找交点，通过边界找坐标值
        if ((LEFT & code) != 0) {
            x = XL;
            y = y1 + (y2 - y1) * (XL - x1) / (x2 - x1);
        }
        else if ((RIGHT & code) != 0) {
            x = XR;
            y = y1 + (y2 - y1) * (XR - x1) / (x2 - x1);
        }
        else if ((BOTTOM & code) != 0) {
            y = YB;
            x = x1 + (x2 - x1) * (YB - y1) / (y2 - y1);
        }
        else if ((TOP & code) != 0) {
```

```
            y = YT;
            x = x1 + (x2 - x1) * (YT - y1) / (y2 - y1);
        }
        //更新范围内的交点
        if (code == code1) {
            x1 = x;
            y1 = y;
            encode(x, y, &code1);
        }
        else {
            x2 = x;
            y2 = y;
            encode(x, y, &code2);
        }
    }

    //最终端点是x1 y1, x2 y2
    //由于本题的特殊性，一定有一个是原始点，那先在一开始记录下
    //把求得的交点push到keep数组中
    //记录原始点 float x10 = x1, y10 = y1, x20 = x2, y20 = y2;
    if (x10 == x1 && y10 == y1) {
        //1点和原始点一样，那2点是交点，存入2点
        keepx[outlen] = x2;
        keepy[outlen] = y2;
        outlen++;//???换个名字就可以全局了
    }
    else if (x20 == x2 && y20 == y2) {
        //2点和原始点一样，那1点是交点，存入1点
        keepx[outlen] = x1;
        keepy[outlen] = y1;
        outlen++;
    }

    //画线
    setlinecolor(YELLOW);
    line(x1, y1, x2, y2);
    return 0;
}

int main() {
    initgraph(640, 480);

    //绘制自定的point多边形
    setcolor(WHITE);
    polygon(Vertex, inlen);
    setcolor(RED);
    polygon(Edge, 4);

    //寻找第一个从外部->内部的边，求交点
    int startnum = -1, endnum = -1;
    for (int i = 0; i < inlen; i++) {
        int code1 = -1, code2 = -1;
        //非末尾点判断
        if (i < inlen - 1) {
            encode(Vertex[i].x, Vertex[i].y, &code1);
            encode(Vertex[i + 1].x, Vertex[i + 1].y, &code2);
```

```
            if (code1 != 0 && code2 == 0) {
                //external to internal
                startnum = i;
                endnum = i + 1;
                break;
            }
        }
        //末尾点判断
        else {
            encode(Vertex[inlen - 1].x, Vertex[inlen - 1].y, &code1);
            encode(Vertex[0].x, Vertex[0].y, &code2);
            if (code1 != 0 && code2 == 0) {
                //external to internal
                startnum = i;
                endnum = i + 1;
                break;
            }
        }
    }
    //看看框和多边形是否有交集
    if (startnum == -1) {
        //没找到交点
        return 0;
    }
    /*else {
        cout << startnum << " " << endnum; //输出判断
    }*/

    //设置全局count不行，给到主函数然后传参进去吧
    //int count = 0;

    //CSCLIP救我老命啊，直接求内外交点然后clip掉
    CSLineClip(Vertex[startnum].x, Vertex[startnum].y, Vertex[endnum].x,
Vertex[endnum].y);

    int curnum = endnum;
    int times = 1;//判断是否循环结束
    while (1) {
        int code1 = -1, code2 = -1;
        int flag = -1;
        //给起始点编码，看看是否在框内
        if (curnum < inlen - 1) {
            encode(Vertex[curnum].x, Vertex[curnum].y, &code1);
            encode(Vertex[curnum + 1].x, Vertex[curnum + 1].y, &code2);
            flag = 1;
        }
        else {
            encode(Vertex[inlen - 1].x, Vertex[inlen - 1].y, &code1);
            encode(Vertex[0].x, Vertex[0].y, &code2);
            flag = 0;
        }

        //两个点都在框内
        if (code1 == 0 && code2 == 0) {
            setlinecolor(YELLOW);
            if (flag == 1) {
```

```
                line(Vertex[curnum].x, Vertex[curnum].y, Vertex[curnum + 1].x,
Vertex[curnum + 1].y);
            }
            else if (flag == 0) {
                line(Vertex[inlen - 1].x, Vertex[inlen - 1].y, Vertex[0].x,
Vertex[0].y);
            }
        }
        //起点内，终点外
        else if (code1 == 0 && code2 != 0) {
            if (flag == 1) {
                CSLineClip(Vertex[curnum].x, Vertex[curnum].y, Vertex[curnum + 1].x,
Vertex[curnum + 1].y);
            }
            else if (flag == 0) {
                CSLineClip(Vertex[inlen - 1].x, Vertex[inlen - 1].y, Vertex[0].x,
Vertex[0].y);
            }
        }
        //起点外，终点内
        else if (code1 != 0 && code2 == 0) {
            if (flag == 1) {
                CSLineClip(Vertex[curnum].x, Vertex[curnum].y, Vertex[curnum + 1].x,
Vertex[curnum + 1].y);
            }
            else if (flag == 0) {
                CSLineClip(Vertex[inlen - 1].x, Vertex[inlen - 1].y, Vertex[0].x,
Vertex[0].y);
            }
        }
        //起点外，终点外
        else {
            //不画线
        }

        //记录入框交点和出框交点，并连结


        //当前线段判断完并连线完，给下一轮循环更新curnum
        if (curnum == inlen - 1) {
            curnum = 0;
            times++;
        }
        else {
            curnum++;
            times++;
        }

        //判断循环退出
        if (times == inlen) {
            break;
        }
    }

    for (int i = 0; i < outlen; i += 2) {
        line(keepx[i], keepy[i], keepx[i + 1], keepy[i + 1]);
```

```
    }

    _getch();                   // 按任意键继续
    closegraph();           // 关闭绘图窗口
    //cout << outlen;
    /*
    //交点判断
    for (int i = 0; i < outlen; i++) {
        cout << keepx[i] << " " << keepy[i] << endl;
    }
    */
    return 0;
}
```