实验二　基于窗口的 Liang-Barsky 的二维直线段裁剪算法的实现

严禁抄袭！

No plagiarism!

一．实验目的

1、学习 Liang-Barsky 直线裁剪算法。
2、实现基于窗口的直线裁剪。

二．实验工具与设备

Clion，MacBook Pro

三、实验内容

1、在本科学习平台（s.ecust.edu.cn）资料栏 lineClipping 文件夹下，有 2 个文件 lineClipping.cpp 和 lineClipping.h。
①在 lineClipping.h 头文件中，定义了点类型 point 和矩形类型 rect
②在 lineClipping.cpp 源文件中，定义了梁友栋-Barsky 直线裁剪算法基础代码

```
int Clip_Top(float p,float q,float &umax,float &umin);
void Line_Clipping(vector<point> &points,rect & winRect);
```

2、在实验一的基础上，利用键盘橡皮筋技术交互绘制要裁剪的直线段，键盘’p’确定直线段端点；利用鼠标橡皮筋技术交互绘制裁剪窗口，鼠标左键单击确定裁剪窗口主对角线位置。

```
rect winObj;                    //标准矩形裁剪窗口对象
int iKeyPointNum = 0;   //键盘'p'确定直线段端点的数目：0-无、1-起始点、2-终止点
int xKey=0,yKey=0;                          //直线段橡皮筋时，保留鼠标移动时的坐标值
int iMousePointNum= 0;//鼠标单击确定裁剪窗口点的数目：0-无、1-起始点、2-终止点
int xMouse=0,yMouse=0;                  //裁剪窗口橡皮筋时，保留鼠标移动时的坐标值
```

3、键盘‘c'实现基于窗口的直线裁剪，观察 Liang-Barsky 直线裁剪算法对直线的裁剪结果。

lineClipping.h:

```
#include <iostream>
#include <vector>
#include <GLUT/glut.h>
```

```
using namespace std;
//点类型 point
typedef struct Point {
   int x, y;
   Point(int a = 0, int b = 0)
   {
      x = a, y = b;
   }
} point;
//矩形类型 rect
typedef struct Rectangle{
   float w_xmin,w_ymin;
   float w_xmax,w_yman;
   Rectangle(float xmin = 0.0, float ymin = 0.0,float xmax=0.0,float yman=0.0){
      w_xmin = xmin; w_ymin = ymin;
      w_xmax = xmax; w_yman = yman;
   }
}rect;
int Clip_Top(float p,float q,float &umax,float &umin);
void Line_Clipping(vector<point> &points,rect & winRect);
```

lineClipping.cpp:

```
#include <vector>
using namespace std;
#include "lineClipping.h"
/*********************************************
*如果 p 参数<0，计算、更新 umax，保证 umax 是最大 u 值
*如果 p 参数>0，计算、更新 umin，保证 umin 是最小 u 值
*如果 umax>umin，返回 0，否则返回 1
*********************************************/
int Clip_Top(float p,float q,float &umax,float &umin){
   float r=0.0;
    if(p<0.0)  //线段从裁剪窗口外部延伸到内部，取最大值 r 并更新 umax
   {
      r=q/p;
      if (r>umin) return 0;  //umax>umin 的情况，弃之
      else if (r>umax)  umax=r;
   }
   else if(p>0.0)      //线段从裁剪窗口内部延伸到外部,取最小值 r 并更新 umin
      {
         r=q/p;
         if (r<umax) return 0;  //umax>umin 的情况，弃之
         else if(r<umin)  umin=r;
      }
      else       //p=0 时，线段平行于裁剪窗口
```

```
        if(q<0.0) return 0;
    return 1;
}
/*************************************************************
*已知 winRect：矩形对象，存放标准裁剪窗口 4 条边信息
*    points：点的动态数组，存放直线 2 个端点信息
*根据裁剪窗口的左、右边界，求 umax;
*根据裁剪窗口的下、上边界，求 umin
*如果 umax>umin，裁剪窗口和直线无交点，否则求裁剪后直线新端点
*************************************************************/
void Line_Clipping(vector<point> &points,rect & winRect){
   //比较左、右边界，获得最大的 umax
   point &p1=points[0],&p2=points[1];
   float dx=p2.x-p1.x,dy=p2.y-p1.y,umax=0.0,umin=1.0;
   point p=p1;
   if (Clip_Top(-dx,p1.x- winRect.w_xmin,umax,umin))  //左边界
      if (Clip_Top(dx,winRect.w_xmax-p1.x, umax,umin)) //右边界
         //比较下、上边界，获得最小的 umin
         if (Clip_Top(-dy,p1.y- winRect.w_ymin, umax,umin)) //下边界
            if (Clip_Top(dy,winRect.w_yman-p1.y, umax,umin)){//求裁剪后直线新端点
               p1.x=(int)(p.x+umax*dx);
               p1.y=(int)(p.y+umax*dy);
               p2.x=(int)(p.x+umin*dx);
               p2.y=(int)(p.y+umin*dy);
            }
}
int count_point=0;
void Line_Clipping_expand(vector<point> &points, rect & winRect,int i){
    //比较左、右边界，获得最大的 umax
    point &p1=points[i],&p2=points[i+1];
    float dx=p2.x-p1.x,dy=p2.y-p1.y,umax=0.0,umin=1.0;
    point p=p1;
    if (Clip_Top(-dx,p1.x- winRect.w_xmin,umax,umin))  //左边界
       if (Clip_Top(dx,winRect.w_xmax-p1.x, umax,umin)) //右边界
          //比较下、上边界，获得最小的 umin
          if (Clip_Top(-dy,p1.y- winRect.w_ymin, umax,umin)) //下边界
             if (Clip_Top(dy,winRect.w_yman-p1.y, umax,umin)){//求裁剪后直线新
端点
                glBegin(GL_LINE_STRIP);//画折线
                glVertex2i((int)(p.x+umax*dx), (int)(p.y+umax*dy));
                glVertex2i((int)(p.x+umin*dx), (int)(p.y+umin*dy));
                glEnd();
             }
}
```

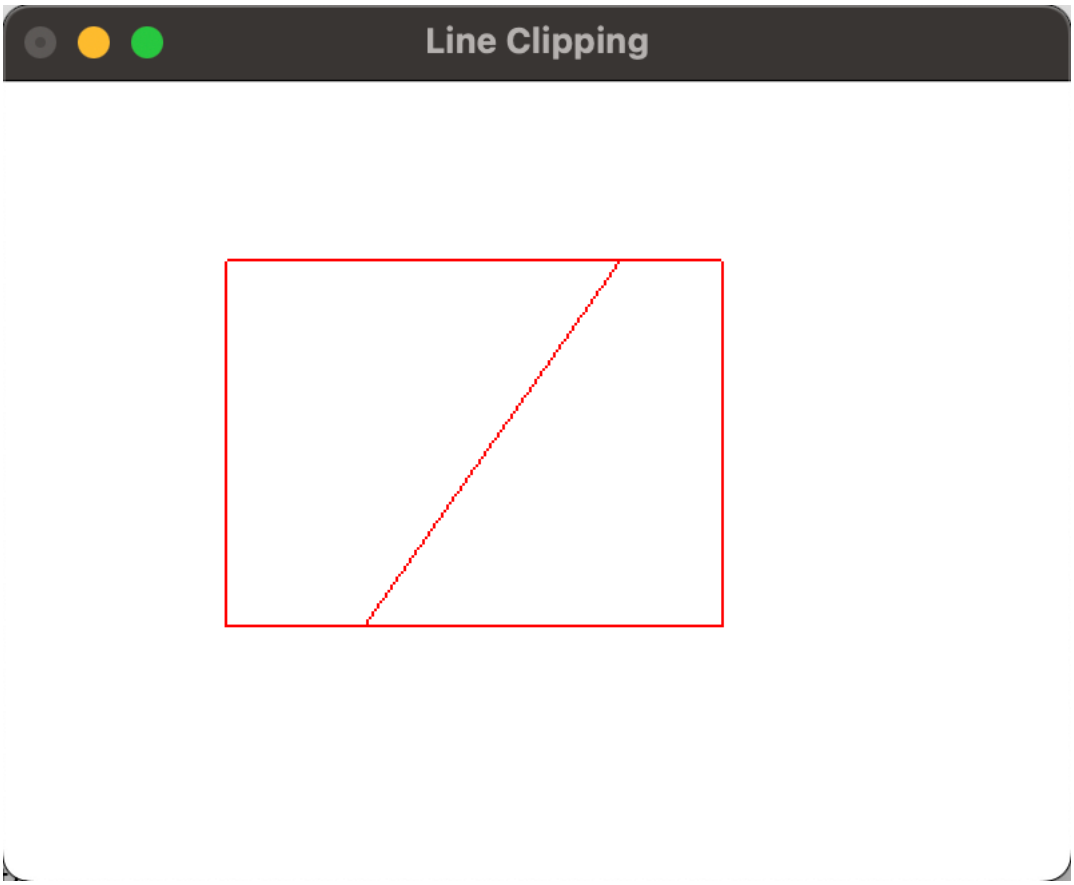
Experiment_2.cpp:

```
#include <GLUT/glut.h>//在 windows 系统上运行请修改 glut 存储路径
#include <vector>
#include "lineClipping.cpp"
using namespace std;

rect obj;
int iKeyNum_p = 0;
int iKeyNum_c = 0;
int iPointNum = 0;
vector<point> points(2);
int x1_rect=0,x2_rect=0,y1_rect=0,y2_rect=0;//矩形对角线的元素

int a0=0,a1=0;
int x1=0,x2=0,y1=0,y2=0;
int pos_x = 0,pos_y = 0;
int judge;
int winWidth = 400, winHeight = 300;
void Initial(void){
    glClearColor(1.0f,1.0f,1.0f,1.0f);
}
void ChangeSize(int w,int h){
    winWidth = w;
    winHeight = h;
    glViewport(0,0,w,h);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluOrtho2D(0.0,winWidth,0.0,winHeight);
}

void Display(void){
    glClear(GL_COLOR_BUFFER_BIT);
    glColor3f(1.0f,0.0f,0.0f);
    if (iKeyNum_p != 0 && iKeyNum_c ==0) {
        glBegin(GL_LINE_STRIP);//start to draw something
        glVertex2i(x1, y1);
        glVertex2i(x2,y2);
        glEnd();
    }
    if (iPointNum != 0 && iKeyNum_c ==0) {
        glBegin(GL_LINE_STRIP);//start to draw something
        glVertex2i(x1_rect, y1_rect);
        glVertex2i(x2_rect, y1_rect);
        glVertex2i(x2_rect, y2_rect);
```

```
        glVertex2i(x1_rect, y2_rect);
        glVertex2i(x1_rect, y1_rect);
        glEnd();
    }
    if(iKeyNum_c == 1){
        points[0].x = x1;
        points[0].y = y1;
        points[1].x = x2;
        points[1].y = y2;
        if(x1_rect<x2_rect){
            obj.w_xmin = x1_rect;
            obj.w_xmax = x2_rect;
        }
        else{
            obj.w_xmin = x2_rect;
            obj.w_xmax = x1_rect;
        }
        if(y1_rect<y2_rect){
            obj.w_ymin = y1_rect;
            obj.w_yman = y2_rect;
        }
        else{
            obj.w_ymin = y2_rect;
            obj.w_yman = y1_rect;
        }
        Line_Clipping(points,obj);
        glBegin(GL_LINE_STRIP);//start to draw something
        glVertex2i(points[0].x, points[0].y);
        glVertex2i(points[1].x,points[1].y);
        glEnd();
        glBegin(GL_LINE_STRIP);//start to draw something
        glVertex2i(x1_rect, y1_rect);
        glVertex2i(x2_rect, y1_rect);
        glVertex2i(x2_rect, y2_rect);
        glVertex2i(x1_rect, y2_rect);
        glVertex2i(x1_rect, y1_rect);
        glEnd();
    }
    glutSwapBuffers();
}

void MousePlot(GLint button, GLint action, GLint xMouse,GLint yMouse){
    if(button == GLUT_LEFT_BUTTON && action == GLUT_DOWN){
        if(iPointNum == 0 || iPointNum == 2){
```

```
            iPointNum = 1;
            x1_rect = xMouse;
            y1_rect = winHeight - yMouse;
        }
        else(iPointNum = 2);
        x2_rect = xMouse;
        y2_rect = winHeight - yMouse;
        glutPostRedisplay();
    }

}
void PassiveMouseMove(GLint xMouse,GLint yMouse){
    if(iKeyNum_p == 1){
        x2 = xMouse;
        y2 = winHeight - yMouse;
        glutPostRedisplay();
    }
    if(iPointNum == 1){
        x2_rect = xMouse;
        y2_rect = winHeight - yMouse;
        glutPostRedisplay();
    }
}
void Key(unsigned char key,GLint xMouse,GLint yMouse){
    if(key == 'p'){
        if(iKeyNum_p == 0 || iKeyNum_p == 2){
            iKeyNum_p = 1;
            x1 = xMouse;
            y1 = winHeight - yMouse;
        }
        else(iKeyNum_p = 2);
        x2 = xMouse;
        y2 = winHeight - yMouse;
        glutPostRedisplay();
    }
    if(key =='c'){
        iKeyNum_c += 1;
        glutPostRedisplay();
    }
}

int main(int argc,char *argv[]){
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB); //初始化窗口的显示模式
```

```
    glutInitWindowSize(400, 300);
    glutInitWindowPosition(100, 100);
    glutCreateWindow("Line Clipping");
    glutDisplayFunc(Display);
    glutMouseFunc(MousePlot);
    glutPassiveMotionFunc(PassiveMouseMove);//动态鼠标
    glutKeyboardFunc(Key);//键盘
    glutReshapeFunc(ChangeSize);
    Initial();
    glutMainLoop();
}
```



四、拓展实验

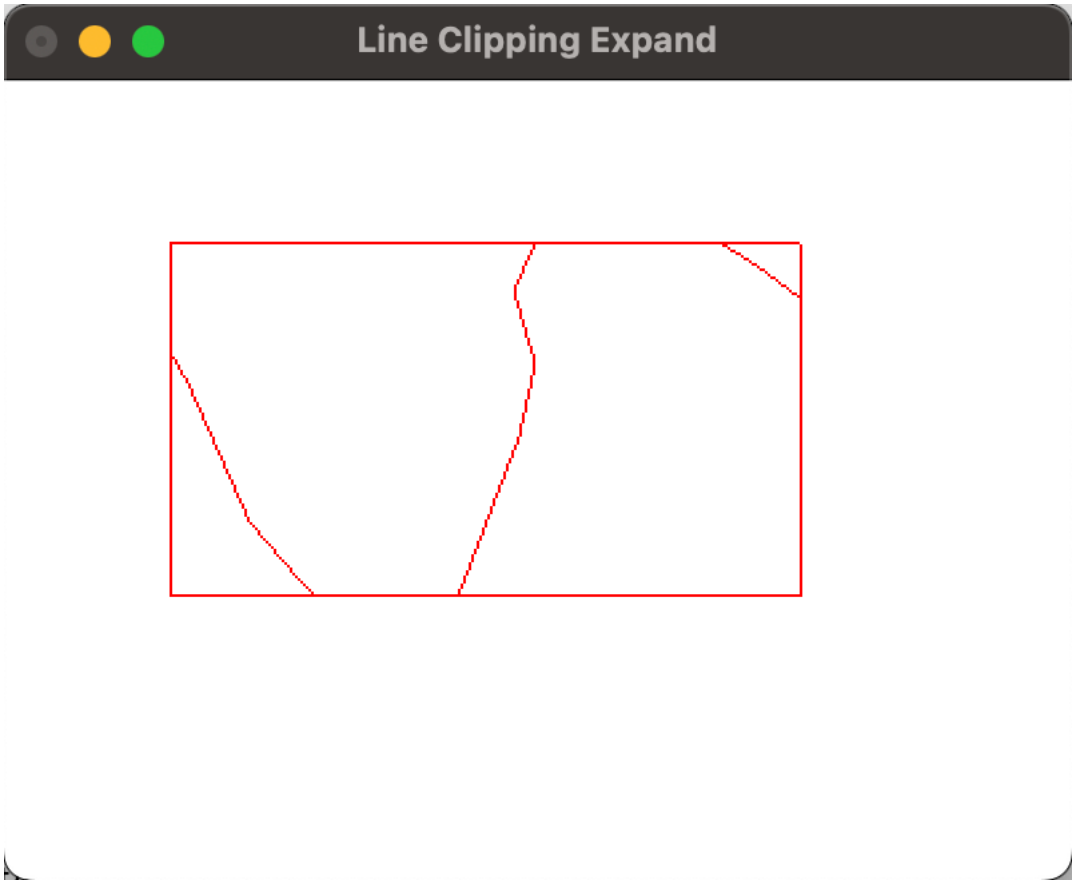
1、举一反三，观察 Liang-Barsky 直线裁剪算法对折线的裁剪结果，利用向量（Vector）保存折线端点序列；鼠标移动交互显示下一段折线的橡皮筋效果，键盘'p'确定折线端点，键盘'e'结束折线绘制。利用鼠标橡皮筋技术交互确定裁剪窗口，键盘'c'实现基于窗口的折线裁剪。

```
#include <GLUT/glut.h>//在 windows 系统上运行请修改 glut 存储路径
#include <vector>
#include "lineClipping.cpp"
using namespace std;
```

```
rect obj;
int iKeyNum_p = 0;
int iKeyNum_c = 0;
int iPointNum = 0;
vector<point> points(100);
vector<point> point0(100);
int x1_rect=0,x2_rect=0,y1_rect=0,y2_rect=0;//矩形对角线的元素
int countp = 0,counte=0;
int x1=0,x2=0,y1=0,y2=0;//端点坐标
int winWidth = 400, winHeight = 300;
void Initial(void){
    glClearColor(1.0f,1.0f,1.0f,1.0f);
}
void createrect(rect &obj){
    if(x1_rect<x2_rect){
        obj.w_xmin = x1_rect;
        obj.w_xmax = x2_rect;
    }
    else{
        obj.w_xmin = x2_rect;
        obj.w_xmax = x1_rect;
    }
    if(y1_rect<y2_rect){
        obj.w_ymin = y1_rect;
        obj.w_yman = y2_rect;
    }
    else{
        obj.w_ymin = y2_rect;
        obj.w_yman = y1_rect;
    }
}
void ChangeSize(int w,int h){
    winWidth = w;
    winHeight = h;
    glViewport(0,0,w,h);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluOrtho2D(0.0,winWidth,0.0,winHeight);
}

void Display(void){
    glClear(GL_COLOR_BUFFER_BIT);
    glColor3f(1.0f,0.0f,0.0f);
    if (iKeyNum_c ==0) {
```

```
        glBegin(GL_LINE_STRIP);//画方框
        glVertex2i(x1_rect, y1_rect);
        glVertex2i(x2_rect, y1_rect);
        glVertex2i(x2_rect, y2_rect);
        glVertex2i(x1_rect, y2_rect);
        glVertex2i(x1_rect, y1_rect);
        glEnd();
        for(int i=0;i<countp;i++){
            glBegin(GL_LINE_STRIP);//画折线
            glVertex2i(points[i].x, points[i].y);
            glVertex2i(points[i+1].x, points[i+1].y);
            glEnd();
        }
    }
    else if(iKeyNum_c == 1){
        createrect(obj);
        for(int i=0;i<countp;i++){
            Line_Clipping_expand(points,obj,i);
        }
        glBegin(GL_LINE_STRIP);//绘制矩形
        glVertex2i(x1_rect, y1_rect);
        glVertex2i(x2_rect, y1_rect);
        glVertex2i(x2_rect, y2_rect);
        glVertex2i(x1_rect, y2_rect);
        glVertex2i(x1_rect, y1_rect);
        glEnd();
    }
    glutSwapBuffers();
}

void MousePlot(GLint button, GLint action, GLint xMouse,GLint yMouse){
    if(button == GLUT_LEFT_BUTTON && action == GLUT_DOWN){
        if(iPointNum == 0 || iPointNum == 2){
            iPointNum = 1;
            x1_rect = xMouse;
            y1_rect = winHeight - yMouse;
        }
        else(iPointNum = 2);
        x2_rect = xMouse;
        y2_rect = winHeight - yMouse;
        glutPostRedisplay();
    }

}
```

```
void PassiveMouseMove(GLint xMouse,GLint yMouse){
    if(counte == 0){
        points[countp].x = xMouse;
        points[countp].y = winHeight - yMouse;
        glutPostRedisplay();
    }
    if(iPointNum == 1){
        x2_rect = xMouse;
        y2_rect = winHeight - yMouse;
        glutPostRedisplay();
    }
}
void Key(unsigned char key,GLint xMouse,GLint yMouse){
    if(key == 'p' && counte == 0) {
        points[countp].x = xMouse;
        points[countp].y = winHeight - yMouse;
        countp++;
        glutPostRedisplay();
    }
    if(key =='c'){
        iKeyNum_c += 1;
        glutPostRedisplay();
    }
    if(key == 'e'){
        counte++;
    }
}

int main(int argc,char *argv[]){
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB); //初始化窗口的显示模式
    glutInitWindowSize(400, 300);
    glutInitWindowPosition(100, 100);
    glutCreateWindow("Line Clipping Expand");
    glutDisplayFunc(Display);
    glutMouseFunc(MousePlot);
    glutPassiveMotionFunc(PassiveMouseMove);//动态鼠标
    glutKeyboardFunc(Key);//键盘
    glutReshapeFunc(ChangeSize);
    Initial();
    glutMainLoop();
}
```



五．思考题

1．Liang-Barsky 的核心思想是什么？

把二维裁剪的问题化成二次一维裁剪问题，而把裁剪问题转化为解一组不等式的问题；