* **截至2023-4-14,18:00，作业情况如下：**

作业1：已完成59人

作业2：已完成57人

作业3：已完成54人

作业4：已完成55人

上机报告：已完成55人

**注：不交算0分，迟交扣分**

## 作业1：模仿课本中点BH算法原理，推导当斜率k>1时的直线扫描转换算法。要求写出推导过程（包括直线方程函数、误差项及其判别、推导关系、初始值，化整等）

【讲解：】Bresenham直线生成算法的基本原理是：每次在最大位移方向上走一步，而在另一方向上根据误差项判别来决定是否走一步。主要步骤：1）根据方程列函数，以及位置判断依据；2）误差项递推推导；3）尽可能化整。

1）对于给定直线段的两个端点*P0(X0, Y0)*和*P1(X1, Y1)*，根据直线方程建立判别函数：

对于直线上的点，*F(x, y)*=0；对于直线上方的点，*F(x, y)*>0；对于直线下方的点，*F(x, y)*<0。

2）当*k*>1时，假设*X0<X1*且*Y0<Y1*，*y*是最大位移方向，因此每次在*y*方向上加1，而在*x*方向加1或不加。假定当前像素点为*Pi(xi, yi)*，则下一候选像素点*Pi+1(xi+1, yi+1)*是在左点*PL(xi, yi+*1*)*或右点*PR(xi+*1*, yi+*1*)*中选一个，设*Mi(xi+*0.5*, yi+*1*)*为*PL*和*PR*的中点。将*Mi*代入F(*x, y*)得到误差项：



当*di*<0时，*Mi*在直线下方，直线离*PL*更近，故*Pi+*1*(xi+*1*, yi+*1*)*取*PL(xi, yi+*1*)*。则*Pi+*1后的中点为*Mi+1(xi+*0.5*, yi+*2*)，*将它代入F(*x, y*)得到误差项

当*di*≥0时，*Mi*在直线上方，直线离*PR*更近，故*Pi+*1*(xi+*1*, yi+*1*)*取*PR(xi+*1*, yi+*1*)*。则*Pi+*1后的中点为*Mi+1(xi+*1.5*, yi+*2*)，*将它代入F(*x, y*)得到误差项

因*P0(X0,Y0)*在直线上，由*M0(x0+*0.5*, y0+*1*)*得出误差项的初始值：

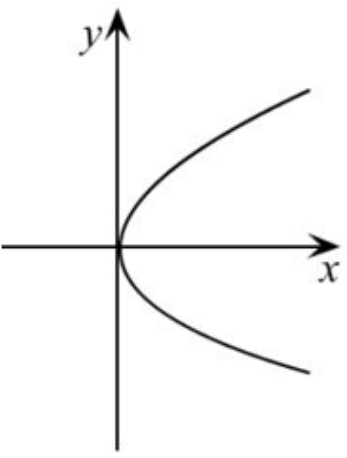
3）用代入化整，即有：

4）综上所述，斜率k>1时的直线扫描转换算法程序如下：（略☺）

## 作业2：模仿第一象限椭圆弧段扫描转换的BH算法，试着用中点Bresenham算法原理推导抛物线 100*x = y2* 的绘制算法。

【讲解：】

更普遍地，推导基于中点Bresenham算法的抛物线 *a****x = y2***的扫描转换方法：



切线

法线N

*P(x, y)*

构建抛物线函数，设x∈[0, xend]。由于该抛物线是关于*x*轴对称的，因此只需计算第一象限中抛物线的像素点位置，另一象限中的像素点可由对称性得到。【注：也可用 F(x,y)=ax-y2则点在抛物线内外的判断等等也要相应调整】

在第一象限，对于在抛物线上的点，有F(*x, y*)=0；对于在抛物线开口内的点，有F(*x, y*)<0；对于在抛物线开口外的点，有F(*x, y*)>0。对于抛物线上任一点*P(x, y)*的法向量为：

得出切线的斜率 （注：对于抛物线的切线斜率也可直接用y’算出）。因此，在2*y<=a*即第一象限抛物线左下部分，*y*为最大位移方向，此时每次在*y*方向上加1，在*x*方向加1或加0；在2*y>a*即第一象限抛物线右上部分，*x*为最大位移方向，此时每次在*x*方向上加1，在*y*方向加1或加0。下面对抛物线在这两部分分别进行算法推导。

（一）在第一象限的左下部分

如下图，此时*y*为最大位移方向。假定当前与抛物线最近的像素点已确定为*Pi(xi, yi)*，则下一候选像素点*Pi+1(xi+1, yi+1)*是左点*PL(xi, yi+1)*或右点*PR(xi+1, yi+1)*，设*Mi(xi+0.5, yi+1)*为*PL*和*PR*的中点。将*Mi*代入F(*x, y*)得到误差项

当*di* ≤0时，中点*Mi*在抛物线开口内，*Pi+1*取*PL*。由*Pi+1*后的中点*Mi+1(xi+0.5, yi+2)*得到误差项

当*di* >0时，中点*Mi*在抛物线开口外，*Pi+1*取*PR*。由*Pi+1*后的中点*Mi+1(xi+1.5, yi+2)*得到误差项

抛物线在此段内的起点为*P0(0, 0)*，则由中点*M0(0.5,1)*算得误差项的初始值

将误差项化整：设*Di = 2di*，则。当*Di* ≤0时，；当*Di* >0时，。



（二）在第一象限的右上部分

如下图，此时*x*为最大位移方向，则*Pi(xi, yi)*下一候选像素点*Pi+1(xi+1, yi+1)*是下点*PD(xi+1, yi)*或上点*PU(xi+1, yi+1)*，设*Mi(xi+1, yi+0.5)*为*PD*和*PU*的中点。将*Mi*代入F(*x, y*)得到误差项

当*di* ≤0时，中点*Mi*在抛物线开口内，*Pi+1*取*PU*。由*Pi+1*后的中点*Mi+1(xi+2, yi+1.5)*得到误差项

当*di* >0时，中点*Mi*在抛物线开口外，*Pi+1*取*PD*。由*Pi+1*后的中点*Mi+1(xi+2, yi+0.5)*得到误差项

将抛物线在左下部分的最后一个像素点作为在此段内的起点为*P0(xe, ye)*，则由中点*M0(xe+1,ye+0.5)*算得误差项的初始值

将误差项化整：设*Di = di*-*0.25*，则。*di* ≤0即*Di* ≤-0.25，因*Di*为整数，故等价于*Di* <0。因此，当*Di* <0时，；当*Di* ≥0时，。



综上所述，基于中点Bresenham算法的抛物线 *a****x = y2***的扫描转换程序为：

void MidBhparabola(int a, int xend, int color) {

int x=0, y=0, d=2-a;

putpixel(x, y, color);

putpixel(x, -y, color);

while (2\*y<=a && x<xend) { // 第一象限左下部分

if(d>0) {

d += 4\*y+6-2\*a;

x++;

y++;

}

else {

d += 4\*y+6;

y++;

}

putpixel(x, y, color);

putpixel(x, -y, color);

}

d = y\*y+y-a\*(x+1); // 第一象限右上部分

while(x<xend){

if(d>=0) {

d -= a;

x++;

}

else {

d += 2\*y+2-a;

x++;

y++;

}

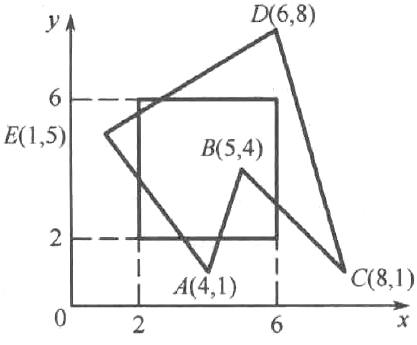
putpixel(x, y, color);

putpixel(x, -y, color);

}

}

## 作业3：课本p172-T6.15：试用Sutherland-Hodgeman多边形裁剪算法对如图所示的多边形进行裁剪，要求画出每次裁剪对应的图形，并标明输入和输出的顶点（类似p167的图6-31，写清过程）。



【讲解：】



根据SH逐边裁剪算法原理，针对窗口的一条边处理多边形的四种情况，依次用窗口的左、下、右、上边界对多边形进行裁剪，其裁剪过程为：

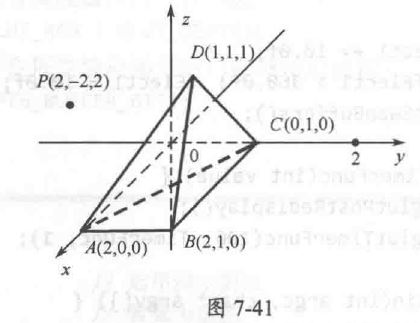
 

即，最终的裁剪结果如下图所示的多边形91234B78



## 作业4：课本p214-T7.7：试作出图7-41中空间四面体的三视图，要求写清变换式（设平移矢量均为1）



【讲解：】

1）主视图：投影变换矩阵为，四面体ABCD到主视图的投影变换齐次坐标计算为：

注：也可用主视图齐次坐标计算形式分别计算各点的投影点坐标

2）俯视图：投影变换矩阵为，四面体ABCD到俯视图的投影变换齐次坐标计算为：

注：也可用俯视图齐次坐标计算形式分别计算各点的投影点坐标

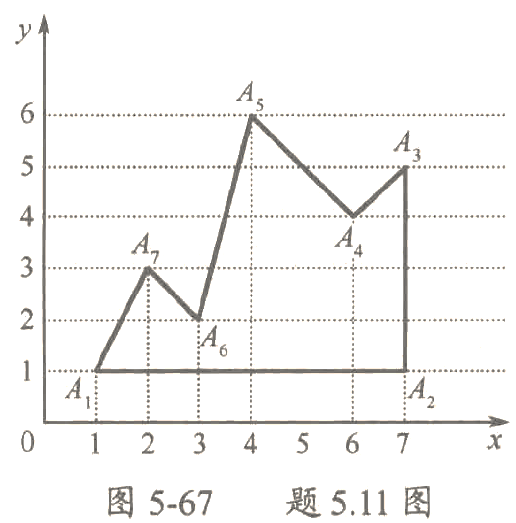
3）侧视图：投影变换矩阵为，四面体ABCD到侧视图的投影变换齐次坐标计算为：

注：也可用侧视图齐次坐标计算形式分别计算各点的投影点坐标

综上计算，四面体ABCD的三视图如下图所示：



## 上机报告：课本p142-T5.11，对如图的多边形采用扫描转换算法（改进的有效边表算法）进行填充，试写出该多边形的ET表和当扫描线y=4时的有效边表AET。



【讲解：】该多边形的ET表如下：



当y=4时的AET表如下：



该多边形的扫描结果如下图所示：（略）