**6、试编写可以对一段任意圆弧进行扫描转换的算法**

 将360度的区域分成8个部分

 3 2

 4 1

 5 8

 6 7

 编写可以对一段任意圆弧进行扫描转换的算法的关键在于，对这段圆弧的起点和终点分别判定是否在同一区域

 如果起点和终点在同一区域，调用中点画圆算法，但要根据实际情况对参数进行修正；

 如果起点和终点不在同一区域，则要根据实际情况对圆弧段进行分割，分割的原则是将每一段的起点和终点放在同一区域，然后分别调用中点画圆算法画圆弧，同样在画的过程中，要根据实际情况对参数进行修正及算法进行修正；

 设圆弧的起点为(x1,y1),终点为(x2,y2)，半径为r

 如图

 A（x1,y1）

 D(x0-r,y0) C (x0,y0)

 B(x2,y2)

将整个圆弧分为两段，弧AC和弧CB，分别进行扫描转换，转换过程中利用中点画圆方法进行，代码如下：

midpoint(x1,y1,x2,y2,r,color,k)

{

 int x,y;

 float d;

 x=x1;

 y=y1;

 d=(x1+1)^2+(y1-0.5)^2-r^2;

 putpixel(x,y,color);

 while (x<=x2)

 {

 if (d<0)

 {

 d+=2\*x+3;

 x++;

 }

 else

 {

 d+=2\*(x-y)+5;

 x++;

 y=y+k;

 }

 }

 putpixel(x,y,color);

 main()

 {

 scanf(“%d”,&n);//分割的圆弧数

 for (i=1;i<=n;i++)

 {

 scanf(“%d,%d,%d,%d,%d”,&x1,&y1,&x2,&y2,,&k); //要求x1<x2

 midpoint(x1,y1,x2,y2,r,color,k);

**7、设计一个多边形区域填充算法，使其边界像素具有一个值，而内部的像素具有另一个值。**

算法设计：

1. 使用画线语句绘制多边形
2. 计算窗口客户区的水平边界最大值MaxX和垂直边界最大值MaxY
3. 调用系统调色板，设置颜色值FillColor为调色板上取得的颜色，CBackColor为白色。
4. 对于每一条边，y从ymin开始，执行下面的循环。
5. x从扫描线和边的交点处开始到窗口客户区右边界，先获得(x, y)位置的像素颜色，如果是填充色，则置成背景色，否则所有填充色填充。执行x=x+1/k，计算下一个x起点值。
6. 如果y=ymin，则扫描结束，否则y++，转（5）。

主要代码：

int MaxX,MaxY;

Void GetMaxX() //求屏幕最大x值

{

 CRect rect;

 GetClientRect(rect);

 MaxX=rect.riht;

}

Void GetMaxY() //求屏幕最大y值

{

 CRect rect;

 GetClientRect(rect);

 MaxX=rect.bottom;

}

Void Draw() //填充多边形函数

{

 COLORREF CBackColor=RGB(255,255,255);//白色

 CClientDC dc(this);

 int m,n,ymin,ymax;

 double x,y,k;

 for (int i=0; i<=6; i++)

 {

 m=i,n=i+1;

 if (7==n) n=0;

 k=(double (Point [m].x- Point [n].x)/ (Point [m].y- Point [n].y);//计算1/k;

 if ((Point [m].y< Point [n].y) //得到每条边y的最大和y最小值

 {

 ymin= Point [m].y;

 ymax= Point [n].y;

 x=Point [m].x; //得到x|ymin

 }

 else

 {

 ymin= Point [n].y;

 ymax= Point [m].y;

 x=Point [n].x;

 }

 For (y=ymin;y<ymax;y++)

{

 For(int j=ROUND(x);j<MaxX;j++)//对每一条扫描线与边的交点的右侧像素循环

 {

 If(dc.GetPixel(j, ROUND(y)==FillColor)//如果像素的颜色是填充色

 {

 dc.SetPixel(j, ROUND(y),CBackColor);//改为背景色

 }

 else

{

 dc.SetPixel(j, ROUND(y),FillColor);//改为填充色

 }

 }

 x+=k; //计算下一个x起点值

 }

 }

}

DrawPolygon() //绘制多边形函数

{

 CClientDC dc(this);

 int m,n;

 for (int j=0;j<=6;j++)

 {

 M=j;n=j+1;

 If (7==n) n=0;

 dc.MoveTo(Point[m]);

 dc.LineTo(Point[n]);

 }

 }

**10、试设计一个生成具有宽度的直线条的算法，使得在直线条连接处不出现图示的缺口**

对于第一种方案：

 c

 d

 e

 a b f

可以把整个线条分成两个部分，

对于左半部分：

先计算出线条的四个顶点，a,b,c,d

再用直线段把相邻角点连接起来，

最后调用多边形填充算法把所得的四边形进行填色

同样，对于右半部分，依法处理

对于第二种方案：

 c

 e

 d f

 g

 a b h

可以把整个线条分成三个部分，

对于左半部分：

先计算出线条的四个顶点，a, b, c, d

再用直线段把相邻角点连接起来，

最后调用多边形填充算法把所得的四边形进行填色

对于中间部分：

先计算出线条的四个顶点，c, d， e，f

再用半径为某个值如R圆弧分别把c, e和 d, f连接起来，

最后调用区域填充算法把所得的圆环段c, e, d, f进行填色

同样，对于右半部分，依法处理

先计算出线条的四个顶点，e, f, g, h

再用直线段把相邻角点连接起来，

最后调用多边形填充算法把所得的四边形进行填色

12、为26个英文大写字母设计5X7的字符掩膜矩阵。

***13、编写一程序实现线段裁剪的中点分割算法***

算法设计：

（1）输入直线段的两端点坐标：P0(x0,y0)，P1(x1,y1)绘制坐标为(wxl, wyt),(wxr,wyb)的窗口

（2）P0点的编码为RC0, P1点的编码为RC1。

（3）若RC0|RC1=0，对直线段应“取”，转步骤（6）；否则若RC0|RC1≠0，对直线段应“弃”，转步骤（6）。

（4）如果直线段有一个端点在窗口内，则计算该直线的中点坐标P[x=(x0+x1)/2,y=(y0+y1)/2]，并计算其编码RC。如果中点P和P0在规定的误差范围内（例如10-6）不重合，则判断中点是否也在窗口内，如果在，PP0 “取”之，否则，PP0 “弃”之。将中点P赋给点P1，转步骤（6）。

（5）如果直线段的两个端点都不在窗口内，则必定与窗口相交，求该直线中点坐标P[x=(x0+x1)/2,y=(y0+y1)/2]和其编码RC。当中点P在窗口外时，如果RC0&RC1=0，则PP1在窗口外，“弃”之；否则PP0在窗口外，“弃”之，直到中点P在窗口内。对于中点在窗口内的直线段PP1和PP0，重复步骤（4）。

（6）输出裁减后的线段。

主要代码：

Void MidClip(double P0x, double P0y, double P1x, double P1y,BOOL flag)

{

 Double x,y;

 Unsigned int RCT0,RCT1;

 RCT0=EnCode(P0x,P0y);

RCT1=EnCode(P1x,P1y);

x=( P0x+ P1x)/2;

y=( P0y+ P1y)/2; RCT=EnCode(x,y);

while(abs(x-P0x)>1e-6|| abs(y-P0y)>1e-6)

{

 if(RCT==0)//中点也在窗口内，则P=P0

 {

 P0x=x;

 P0y=y;

 RCT0= RCT;

 }

else //否则舍弃P1点

 {

 P1x=x;

 P1y=y;

 RCT1= RCT;

 }

 x=( P0x+ P1x)/2;

y=( P0y+ P1y)/2; RCT=EnCode(x,y);

 }

 if(flag==true)

 {

 Pointx[1]=x;

 Pointy[1]=y;

 }

 else

 {

 Pointx[0]=x;

 Pointy[0]=y;

 }

 }

***14 编写一程序实现逐次多边形裁剪算法***

算法设计：

（1）输入第一个顶点坐标：F(x0,y0)

第二个顶点坐标：S(x1,y1)

 (2) 当顶点输入完毕，转（7）

（3）输入顶点P坐标：P(x2,y2)

（4）SP与裁剪线相交吗？是，求SP与裁剪线的交点I（x,y）,并输出I坐标：I(x,y)

（5）P位于可见一侧吗？是，输出顶点P坐标：P(x2,y2)

（6）将顶点P坐标：P(x2,y2)=》顶点S坐标：S(x1,y1)，转（2）

（7）将顶点F坐标：F(x0,y0)=》顶点P坐标：S(x2,y2)，形成闭合，

（8） SP与裁剪线相交吗？是，求SP与裁剪线的交点I（x,y）,并输出I坐标：I(x,y)

（9）P位于可见一侧吗？是，输出顶点P坐标：P(x2,y2)

（10）结束

程序代码：

/\* Sutherland-Hodgman 算法 \*/

#define LEN sizeof(struct node)

#include <math.h>

#include "display.h"

 struct node {

 int dx,dy;

 struct node \*next;

 };

struct node \*creat()

{ struct node \*h,\*q,\*r;

 int p[8][2]={100,120,160,50,180,100,200,80,240,160,210,220,170,160,140,190};

 int i;

 setcolor(12);

 for (i=0;i<7;i++) line(p[i][0],p[i][1],p[i+1][0],p[i+1][1]);

 line(p[0][0],p[0][1],p[7][0],p[7][1]);

 rectangle(120,200,230,70);

 h=NULL;

 for (i=0;i<8;i++)

 { q=(struct node \*)malloc(LEN);

 q->dx=p[i][0]; q->dy=p[i][1];

 if (h==NULL) h=q;

 else r->next=q;

 r=q;

 }

 r->next=NULL;

 return(h);

}

struct node \*builx(h,x)

struct node \*h;

int x;

{int s[2],j[2];

 struct node \*hh,\*p,\*r,\*q;

 int max,min;

 p=h; hh=NULL;

 s[0]=p->dx; s[1]=p->dy;

 p=p->next;

 while (p!=NULL)

 { j[0]=x;

 j[1]=s[1]+(p->dy-s[1])\*(x-s[0])/(p->dx-s[0]);

 max=s[0]; min=p->dx;

 if (s[0]<p->dx) { max=p->dx; min=s[0]; }

 if ((j[0]>=min)&&(j[0]<=max))

 { q=(struct node \*)malloc(LEN);

 q->dx=j[0]; q->dy=j[1];

 if (hh==NULL) hh=q;

 else r->next=q;

 r=q;

 }

 if (p->dx>=x)

 { q=(struct node \*)malloc(LEN);

 q->dx=p->dx; q->dy=p->dy;

 if (hh==NULL) hh=q;

 else r->next=q;

 r=q;

 }

 s[0]=p->dx; s[1]=p->dy;

 p=p->next;

 }

 p=h;

 j[0]=x; j[1]=s[1]+(p->dy-s[1])\*(x-s[0])/(p->dx-s[0]);

 max=s[0]; min=p->dx;

 if (s[0]<p->dx) { max=p->dx; min=s[0]; }

 if ((j[0]>=min)&&(j[0]<=max))

 { q=(struct node \*)malloc(LEN);

 q->dx=j[0]; q->dy=j[1];

 if (hh==NULL) hh=q;

 else r->next=q;

 r=q;

 }

 if (p->dx>=x)

 { q=(struct node \*)malloc(LEN);

 q->dx=p->dx; q->dy=p->dy;

 if (hh==NULL) hh=q;

 else r->next=q;

 r=q;

 }

 r->next=NULL;

 return(hh);

}

struct node \*builxx(h,x)

struct node \*h;

int x;

{int s[2],j[2];

 struct node \*hh,\*p,\*r,\*q;

 int max,min;

 p=h;

 hh=NULL;

 s[0]=p->dx;

 s[1]=p->dy;

 p=p->next;

 while (p!=NULL)

 {

 j[0]=x;

 j[1]=s[1]+(p->dy-s[1])\*(x-s[0])/(p->dx-s[0]+0.1);

 max=s[0];

 min=p->dx;

 if (s[0]<p->dx) { max=p->dx;

 min=s[0];

 }

 if ((j[0]>=min)&&(j[0]<=max))

 { q=(struct node \*)malloc(LEN);

 q->dx=j[0];

 q->dy=j[1];

 if (hh==NULL)

 hh=q;

 else r->next=q;

 r=q;

 }

 if (p->dx<=x)

 { q=(struct node \*)malloc(LEN);

 q->dx=p->dx;

 q->dy=p->dy;

 if (hh==NULL)

 hh=q;

 else r->next=q;

 r=q;

 }

 s[0]=p->dx;

 s[1]=p->dy;

 p=p->next;

 }

 p=h;

 j[0]=x;

 j[1]=s[1]+(p->dy-s[1])\*(x-s[0])/(p->dx-s[0]+0.1);

 max=s[0];

 min=p->dx;

 if (s[0]<p->dx) { max=p->dx;

 min=s[0];

 }

 if ((j[0]>=min)&&(j[0]<=max))

 { q=(struct node \*)malloc(LEN);

 q->dx=j[0];

 q->dy=j[1];

 if (hh==NULL)

 hh=q;

 else r->next=q;

 r=q;

 }

 if (p->dx<=x)

 { q=(struct node \*)malloc(LEN);

 q->dx=p->dx;

 q->dy=p->dy;

 if (hh==NULL)

 hh=q;

 else r->next=q;

 r=q;

 }

 r->next=NULL;

 return(hh);

}

struct node \*buily(h,y)

struct node \*h;

int y;

{int s[2],j[2];

 struct node \*hh,\*p,\*r,\*q;

 int max,min;

 p=h;

 hh=NULL;

 s[0]=p->dx;

 s[1]=p->dy;

 p=p->next;

 while (p!=NULL)

 {

 j[1]=y;

 j[0]=s[0]+(p->dx-s[0])\*(y-s[1])/(p->dy-s[1]+0.1);

 max=s[1];

 min=p->dy;

 if (s[1]<p->dy) { max=p->dy;

 min=s[1];

 }

 if ((j[1]>=min)&&(j[1]<=max))

 { q=(struct node \*)malloc(LEN);

 q->dx=j[0];

 q->dy=j[1];

 if (hh==NULL)

 hh=q;

 else r->next=q;

 r=q;

 }

 if (p->dy>=y)

 { q=(struct node \*)malloc(LEN);

 q->dx=p->dx;

 q->dy=p->dy;

 if (hh==NULL)

 hh=q;

 else r->next=q;

 r=q;

 }

 s[0]=p->dx;

 s[1]=p->dy;

 p=p->next;

 }

 p=h;

 j[1]=y;

 j[0]=s[0]+(p->dx-s[0])\*(y-s[1])/(p->dy-s[1]+0.1);

 max=s[1];

 min=p->dy;

 if (s[1]<p->dy) { max=p->dy;

 min=s[1];

 }

 if ((j[1]>=min)&&(j[1]<=max))

 { q=(struct node \*)malloc(LEN);

 q->dx=j[0];

 q->dy=j[1];

 if (hh==NULL)

 hh=q;

 else r->next=q;

 r=q;

 }

 if (p->dy>=y)

 { q=(struct node \*)malloc(LEN);

 q->dx=p->dx;

 q->dy=p->dy;

 if (hh==NULL)

 hh=q;

 else r->next=q;

 r=q;

 }

 r->next=NULL;

 return(hh);

}

struct node \*builyy(h,y)

struct node \*h;

int y;

{int s[2],j[2];

 struct node \*hh,\*p,\*r,\*q;

 int max,min;

 p=h;

 hh=NULL;

 s[0]=p->dx;

 s[1]=p->dy;

 p=p->next;

 while (p!=NULL)

 {

 j[1]=y;

 j[0]=s[0]+(p->dx-s[0])\*(y-s[1])/(p->dy-s[1]+0.1);

 max=s[1];

 min=p->dy;

 if (s[1]<p->dy) { max=p->dy;

 min=s[1];

 }

 if ((j[1]>=min)&&(j[1]<=max))

 { q=(struct node \*)malloc(LEN);

 q->dx=j[0];

 q->dy=j[1];

 if (hh==NULL)

 hh=q;

 else r->next=q;

 r=q;

 }

 if (p->dy<=y)

 { q=(struct node \*)malloc(LEN);

 q->dx=p->dx;

 q->dy=p->dy;

 if (hh==NULL)

 hh=q;

 else r->next=q;

 r=q;

 }

 s[0]=p->dx;

 s[1]=p->dy;

 p=p->next;

 }

 p=h;

 j[1]=y;

 j[0]=s[0]+(p->dx-s[0])\*(y-s[1])/(p->dy-s[1]+0.1);

 max=s[1];

 min=p->dy;

 if (s[1]<p->dy) { max=p->dy;

 min=s[1];

 }

 if ((j[1]>=min)&&(j[1]<=max))

 { q=(struct node \*)malloc(LEN);

 q->dx=j[0];

 q->dy=j[1];

 if (hh==NULL)

 hh=q;

 else r->next=q;

 r=q;

 }

 if (p->dy<=y)

 { q=(struct node \*)malloc(LEN);

 q->dx=p->dx;

 q->dy=p->dy;

 if (hh==NULL)

 hh=q;

 else r->next=q;

 r=q;

 }

 r->next=NULL;

 return(hh);

}

main()

{

 int max,min;

 struct node \*head,\*r,\*q;

 int i;

 int s[2];

 Initialize();

 head=creat();

 q=head;

 while (q->next!=NULL)

 { putpixel(q->dx,q->dy,14);

 q=q->next;

 }

 putpixel(q->dx,q->dy,14); /\*左边界进行裁减\*/

 q=builx(head,120);

 head=q;

 while (q->next!=NULL)

 { putpixel(q->dx,q->dy,15);

 q=q->next;

 }

 putpixel(q->dx,q->dy,15); /\*上边界进行裁减\*/

 q=buily(head,70);

 head=q;

 while (q->next!=NULL)

 { putpixel(q->dx,q->dy,2);

 q=q->next;

 }

 putpixel(q->dx,q->dy,2); /\*右边界进行裁减\*/

 q=builxx(head,230);

 head=q;

 while (q->next!=NULL)

 { putpixel(q->dx,q->dy,1);

 q=q->next;

 }

 putpixel(q->dx,q->dy,1); /\*下边界进行裁减\*/

 q=builyy(head,200);

 head=q;

 s[0]=q->dx;

 s[1]=q->dy;

 q=q->next;

 setcolor(14);

 outtextxy(100,100,"abc");

 while (q!=NULL)

 { line(s[0],s[1],q->dx,q->dy);

 s[0]=q->dx;

 s[1]=q->dy;

 q=q->next;

 getch();

 }

 q=head;

 outtextxy(200,200,"def ");

 line(s[0],s[1],q->dx,q->dy);

 getch();

}