Drawing the basic primitives

```
#include<stdio.h>
#include<graphics.h>
int main(){
int gd=DETECT,gm;
initgraph(&gd,&gm,(char*)"");
circle(300,100,50);
circle(360,100,10);
circle(240,100,10);
circle(280,100,4);
circle(320,100,4);
line(300,105,300,115);
line(300,115,305,115);
line(280,145,280,170);
line(320,145,320,170);
rectangle(240,170,360,300);
line(240,200,200,240);
line(360,200,400,240);
line(270,300,270,350);
line(330,300,330,350);
line(270,350,250,360);
line(330,350,350,360);
getch();
closegraph();
}
```

DDA

```
#include <graphics.h>
#include <stdio.h>
#include <math.h>
#include <dos.h>
int main(){
float x,y,x1,y1,x2,y2,dx,dy,step;
int i,gd=DETECT,gm;
printf("Enter the value of x1 and y1 \n");
scanf("%f%f",&x1,&y1);
printf("Enter the value of x2 and y2: \n");
scanf("%f%f",&x2,&y2);
initgraph(&gd,&gm,(char*)"");
dx=abs(x2-x1);
dy=abs(y2-y1);
if(dx>=dy)
step=dx;
else
step=dy;
dx=dx/step;
dy=dy/step;
x=x1;
y=y1;
i=1;
while(i<=step){
putpixel(x,y,WHITE);
x=x+dx;
y=y+dy;
i=i+1;
}
getch();
closegraph();
}
```

Bresenham's Line Generation

```
#include<graphics.h>
#include<stdlib.h>
#include<stdio.h>
#include<math.h>
#include<conio.h>
#include<stdio.h>

/****MAIN FUNCTION*****/
void main()
{
int x1,x2,y1,y2,e,x,y,s1,s2,dx,dy,a,xic,yic;
int interchang,i;
int gdriver=DETECT,gmode,errorcode;
initgraph(&gdriver,&gmode,"c:/tc/bgi");
errorcode=graphresult();
if(errorcode!=grOk)
{
printf("GraphicsError:%s\n",grapherrormsg(errorcode));
printf("Press Any Key To Halt");
```



```
getch();
exit(1);
}
printf("Enter the value of x1 and y1:");
scanf("%d%d",&x1,&y1);
printf("Enter the value of x2 and y2:");
scanf("%d%d",&x2,&y2);
dx=abs(x2-x1);
dy=abs(y2-y1);
e=(2*dy)-dx;
x=x1;
```

```
y=y1;
for(i=0;i<=dx;i++)
{ delay(500);
putpixel(x,y,10);
while(e>=0)
{
y=y+1;
e=e-(2*dx);
}
x=x+1;
e=e+(2*dy);
}
getch();
}
```

Boundary Fill Algorithm

```
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
void boundaryfill(int x,int y,int fillcolour,int boundarycolour){
int c;
c=getpixel(x,y);
if((c!=fillcolour)&&(c!=boundarycolour)){
putpixel(x,y,fillcolour);
boundaryfill(x,y,fillcolour,boundarycolour);
boundaryfill(x+1,y,fillcolour,boundarycolour);
boundaryfill(x-1,y,fillcolour,boundarycolour);
boundaryfill(x,y+1,fillcolour,boundarycolour);
boundaryfill(x,y-1,fillcolour,boundarycolour);
boundaryfill(x+1,y+1,fillcolour,boundarycolour);
boundaryfill(x+1,y-1,fillcolour,boundarycolour);
boundaryfill(x-1,y+1,fillcolour,boundarycolour);
boundaryfill(x-1,y-1,fillcolour,boundarycolour);
}
}
int main(){
int gd = DETECT, gm;
initgraph(&gd,&gm,(char*)"");
int x=200,y=250,fillcolour,boundarycolour;
rectangle(100,100,300,300);
boundaryfill(x,y,BLUE,WHITE);
getch();
closegraph();
}
```

Flood Fill Algorithm

```
#include<stdio.h>
#include<graphics.h>
#include<stdlib.h>
void flood(int x, int y,int color1, int color2){
int c;
c=getpixel(x,y);
if(c==0){
putpixel( x,y,color1);
flood(x+1,y,color1,color2);
flood(x-1,y,color1,color2);
flood(x,y+1,color1,color2);
flood(x,y-1,color1,color2);
}
}
int main(){
int gd=DETECT,gm;
initgraph(&gd,&gm,(char*)"");
int x, y,r,color1,color2;
x=200;
y=250;
r=100;
circle(x,y,r);
color1=10;
color2=0;
flood(x,y,color1,color2);
getch();
closegraph();
}
```

2D Transformations translation, scaling, shearing, reflection and rotation.

```
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
#include<math.h>
void obj(int a[4][2]);
void translation();
void rotation();
void reflection();
void shearing();
void scaling();
int a[4][2]={{0,0}
,{40,0}
,{40,40},
{0,40}};
int b[4][2];
int main()
{
int gd=DETECT,gm,n;
initgraph(&gd,&gm,(char*)"");
do{
setcolor(WHITE);
line(0,getmaxy()/2,getmaxx(),getmaxy()/2);
line(getmaxx()/2,0,getmaxx()/2,getmaxy());
setcolor(12);
obj(a);
int ch;
setcolor(YELLOW);
printf("\n1.translation\n2.rotation(90 degree)\n3.scaling\n4.reflection\n5.shearing");
```

```
printf("\nenter your choice\n");
scanf("%d",&ch);
switch(ch){
case 1:
translation();
break;

case 2:
rotation();
break;
case 3:
scaling();
break;
case 4:
reflection();
break;
case 5:
shearing();
break;
}
printf("\nenter 1 to continue or 0 to stop\n");
scanf("%d",&n);
cleardevice();
}while(n==1);
getch();
closegraph();
}
void obj(int s[4][2])
{
```

```
line(getmaxx()/2+s[0][0],getmaxy()/2-s[0][1],getmaxx()/2+s[1][0],getmaxy()/2-s[1][1]);
line(getmaxx()/2+s[1][0],getmaxy()/2-s[1][1],getmaxx()/2+s[2][0],getmaxy()/2-s[2][1]);
line(getmaxx()/2+s[2][0],getmaxy()/2-s[2][1],getmaxx()/2+s[3][0],getmaxy()/2-s[3][1]);
line(getmaxx()/2+s[3][0],getmaxy()/2-s[3][1],getmaxx()/2+s[0][0],getmaxy()/2-s[0][1]);
}
void translation(){
int tx=40,ty=40;
for(int i=0; i<4;i++)
{
b[i][0]=a[i][0]+tx;
b[i][1]=a[i][1]+ty;
}
obj(b);
}
void rotation()
{
int th=90;
for(int i=0; i<4;i++)
{

b[i][0]=a[i][0]*cos(th)-a[i][0]*sin(th);
b[i][1]=a[i][1]*sin(th)+a[i][1]*cos(th);
}
obj(b);
}
void reflection(){
int x,y;
for(int i=0;i<4;i++)
{
```

```
b[i][0]=-a[i][0];
b[i][1]=-a[i][1];
}
obj(b);
}
void scaling(){
int sx,sy;
sx=2;
sy=3;
for(int i=0;i<4;i++)
{
b[i][0]=a[i][0]*sx;
b[i][1]=a[i][1]*sy;
}
obj(b);
}
void shearing(){
int t[2][2]={ {1,1},
{0,1}};
for(int i=0;i<4;i++){
for(int j=0;j<4;j++){
b[i][j]=0;
for(int p=0;p<2;p++){
b[i][j]=b[i][j]+a[i][p]*t[p][j];
}
}
}
obj(b);
}
```

Character Generation

```
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
int main()
{
int i,j;
int gd = DETECT,gm;
initgraph(&gd,&gm,(char*)"");
int O[][10]={{1,1,1,1,1,1,1,1,1,1},

{1,1,1,1,1,1,1,1,1,1},
{1,1,0,0,0,0,0,0,1,1},
{1,1,0,0,0,0,0,0,1,1},
{1,1,0,0,0,0,0,0,1,1},
{1,1,0,0,0,0,0,0,1,1},
{1,1,0,0,0,0,0,0,1,1},
{1,1,0,0,0,0,0,0,1,1},
{1,1,0,0,0,0,0,0,1,1},
{1,1,1,1,1,1,1,1,1,1},
{1,1,1,1,1,1,1,1,1,1}};

int M[][10]={{1,1,0,0,0,0,0,0,1,1},

{1,1,0,0,0,0,0,0,1,1},
{1,1,1,0,0,0,1,1,1,1},
{1,1,1,1,0,0,1,1,1,1},
{1,1,0,1,1,1,0,0,1,1},
{1,1,0,0,1,1,0,0,1,1},
```

```
{1,1,0,0,0,0,0,0,1,1},
{1,1,0,0,0,0,0,0,1,1},
{1,1,0,0,0,0,0,0,1,1},
{1,1,0,0,0,0,0,0,1,1}};

for(i=0;i<10;i++)
{
for(j=0;j<10;j++)
{
if (O[j][i]==1)
{
putpixel(i+200,j+200,YELLOW);
}
if (M[j][i]==1)
{
putpixel(i+220,j+200,YELLOW);
}
}
}
getch();
closegraph();
}
```

Line Clipping Algorithm

```
#include<graphics.h>
#include<dos.h>
#include<conio.h>
#include<stdlib.h>
#include<math.h>

int main()
{ int gd, gm ;
int x1 , y1 , x2 , y2 ;
int wxmin,wymin,wxmax, wymax ;
float u1 = 0.0,u2 = 1.0 ;

int p1 , q1 , p2 , q2 , p3 , q3 , p4 ,q4 ;
float r1 , r2 , r3 , r4 ; int x11 , y11 , x22 , y22 ;
wxmin=100;
wymin=100;
wxmax=200;
wymax=200;
x1=110;
y1=60;
x2=260;
y2=210;
/
p1 = -(x2 - x1 );
q1 = x1 - wxmin ;
p2 = ( x2 - x1 ) ;
q2 = wxmax - x1 ;
p3 = - ( y2 - y1 ) ; q3 = y1 - wymin ;
```

```
p4 = ( y2 - y1 ) ; q4 = wymax - y1 ;
printf("p1=0 line is parallel to left clipping\n");
printf("p2=0 line is parallel to right clipping\n");
printf("p3=0 line is parallel to bottom clipping\n");
printf("p4=0 line is parallel to top clipping\n");

if() || ( ( p4 == 0.0 ) && ( q4 < 0.0 ) ) ) ( ( p1 == 0.0 ) && ( q1 < 0.0 ) ) || ( ( p2 == 0.0 ) && ( q2
< 0.0 ) ) || ( ( p3 == 0.0 ) && ( q3 < 0.0 )
{ printf("Line is rejected\n");
getch();
detectgraph(&gd,&gm);
initgraph(&gd,&gm,(char*)"");
setcolor(RED);
rectangle(wxmin,wymax,wxmax,wymin);
setcolor(BLUE);
line(x1,y1,x2,y2);
getch();
setcolor(WHITE);
line(x1,y1,x2,y2);
getch();
}
else

{
if( p1 != 0.0 )
{ float r12;
r12 =float(q1 /p1);
printf("%f",r12);
r1 =(float) q1 /p1 ;
printf("%f",r1);
```

```
if( p1 < 0 )
u1 = fmax(r1 , u1 );
else
u2 = fmin(r1 , u2 );
}
if( p2 != 0.0 )
{ r2 = (float ) q2 /p2 ;
if( p2 < 0 )
u1 = fmax(r2 , u1 );
else
u2 = fmin(r2 , u2 );

}
if( p3 != 0.0 )
{
r3 = (float )q3 /p3 ;
if( p3 < 0 )
u1 = fmax(r3 , u1 );
else
u2 = fmin(r3 , u2 );
}
if( p4 != 0.0 )
{ r4 = (float )q4 /p4 ;
if( p4 < 0 )

u1 = fmax(r4 , u1 );
else
u2 = fmin(r4 , u2 );
```

```
}

if( u1 > u2 )
printf("line rejected\n");
else
{

x11 = x1 + u1 * ( x2 - x1 ) ;
y11 = y1 + u1 * ( y2 - y1 ) ;

x22 = x1 + u2 * ( x2 - x1 );
y22 = y1 + u2 * ( y2 - y1 );

printf("Original line cordinates\n");
printf("x1 = %d , y1 = %d, x2 = %d, y2 = %d\n",x1,y1,x2,y2);

printf("Windows coordinate are \n");
printf("wxmin = %d, wymin = %d,wxmax = %d , wymax = %d
",wxmin,wymin,wxmax,wymax);

printf("New coordinates are \n");

printf("x11 = %d, y11 = %d,x22 = %d , y22 = %d\n",x11,y11,x22,y22);
detectgraph(&gd,&gm);
initgraph(&gd,&gm,(char*)"");
setcolor(2);
rectangle(wxmin,wymax,wxmax,wymin);
setcolor(1);
line(x1,y1,x2,y2);
//getch();
```

```
//setcolor(0);
//line(x1,y1,x2,y2);
setcolor(12);
line(x11,y11,x22,y22);
//setcolor(BLACK);
//line(70,100,170,200);
getch();
}
}
}
```

Bezier Curve

```
#include <stdio.h>
#include <graphics.h>
#include <math.h>
int x[4]={200,100,200,250};
int y[4]={200,150,75,100};
void bezier ()
{
int i;
float t,ptx,pty;
for (t = 0; t <= 1; t += 0.0001)
{
ptx = pow(1-t,3)*x[0]+3*t*pow(1-t,2)*x[1]+3*pow(t,2)*(1-t)*x[2]+pow(t,3)*x[3];
pty = pow(1-t,3)*y[0]+3*t*pow(1-t,2)*y[1]+3*pow(t,2)*(1-t)*y[2]+pow(t,3)*y[3];
putpixel (ptx, pty,YELLOW);
}
for (i=0; i<4; i++)
putpixel (x[i], y[i], BLUE);
getch();

closegraph();
}
int main()
{
int gd = DETECT, gm;
initgraph (&gd, &gm,(char*)"");
bezier ();
}
```

Cohen sutherland line clipping

```
#include<graphics.h>

#include<conio.h>

#include<stdio.h>

#include<math.h>

void main()

{

int
rcode_begin[4]={0,0,0,0},rcode_end[4]={0,0,0,0},region_code
[4];

int W_xmax,W_ymax,W_xmin,W_ymin,flag=0;

float slope;

int x,y,x1,y1,i, xc,yc;

int gr=DETECT,gm;

initgraph(&gr,&gm,"C:\\TURBOC3\\BGI");

printf("\n****** Cohen Sutherlsnd Line Clipping algorithm
***********");

printf("\n Now, enter XMin, YMin =");



scanf("%d %d",&W_xmin,&W_ymin);

printf("\n First enter XMax, YMax =");

scanf("%d %d",&W_xmax,&W_ymax);

printf("\n Please enter intial point x and y= ");

scanf("%d %d",&x,&y);

printf("\n Now, enter final point x1 and y1= ");

scanf("%d %d",&x1,&y1);
```

```
cleardevice();
rectangle(W_xmin,W_ymin,W_xmax,W_ymax);
line(x,y,x1,y1);
line(0,0,600,0);
line(0,0,0,600);
if(y>W_ymax)  {
rcode_begin[0]=1;     // Top
flag=1 ;
}
if(y<W_ymin) {
rcode_begin[1]=1;           // Bottom
flag=1;
}
if(x>W_xmax)  {
rcode_begin[2]=1;           // Right
flag=1;
}
if(x<W_xmin)   {
rcode_begin[3]=1;           //Left
flag=1;
}

//end point of Line
if(y1>W_ymax){
```

```
rcode_end[0]=1;            // Top
flag=1;
}
if(y1<W_ymin) {
rcode_end[1]=1;            // Bottom
flag=1;
}
if(x1>W_xmax){
rcode_end[2]=1;            // Right
flag=1;
}
if(x1<W_xmin){
rcode_end[3]=1;            //Left
flag=1;
 }
if(flag==0)
{
printf("No need of clipping as it is already in window");
}
flag=1;
for(i=0;i<4;i++){
region_code[i]= rcode_begin[i] && rcode_end[i] ;
if(region_code[i]==1)
 flag=0;
```

```
}
if(flag==0)
{
printf("\n Line is completely outside the window");
}
else{
slope=(float)(y1-y)/(x1-x);
if(rcode_begin[2]==0 && rcode_begin[3]==1)   //left
{
y=y+(float) (W_xmin-x)*slope ;
x=W_xmin;

}
if(rcode_begin[2]==1 && rcode_begin[3]==0)       // right
{
y=y+(float) (W_xmax-x)*slope ;
x=W_xmax;

}
if(rcode_begin[0]==1 && rcode_begin[1]==0)      // top
{
x=x+(float) (W_ymax-y)/slope ;
y=W_ymax;
```

```
}
if(rcode_begin[0]==0 && rcode_begin[1]==1)     // bottom
{
x=x+(float) (W_ymin-y)/slope ;
y=W_ymin;

}
// end points
if(rcode_end[2]==0 && rcode_end[3]==1)   //left
{
y1=y1+(float) (W_xmin-x1)*slope ;
x1=W_xmin;

}
if(rcode_end[2]==1 && rcode_end[3]==0)       // right
{
y1=y1+(float) (W_xmax-x1)*slope ;
x1=W_xmax;

}
if(rcode_end[0]==1 && rcode_end[1]==0)      // top
{
x1=x1+(float) (W_ymax-y1)/slope ;
y1=W_ymax;
```

```
}
if(rcode_end[0]==0 && rcode_end[1]==1)     // bottom
{
x1=x1+(float) (W_ymin-y1)/slope ;
y1=W_ymin;

}
}
delay(1000);
clearviewport();
rectangle(W_xmin,W_ymin,W_xmax,W_ymax);
line(0,0,600,0);
line(0,0,0,600);
setcolor(RED);
line(x,y,x1,y1);
getch();
closegraph();
}
```

Mid Point Circle

```
#include<stdio.h>
#include<stdlib.h>
#include<process.h>
#include<graphics.h>

int main()
{int gd=DETECT,gm;
initgraph(&gd,&gm,(char*)"");
        int cx,cy,r,x,y;
        printf("enter centre coordinates of circle");
        scanf("%d%d",&cx,&cy);
        printf("ENter radius");
        scanf("%d",&r);
        float m;
        x=0;
        y=r;
        m=(5/4)-r;

        while(x<=y)
        {
                if(m<0)
                {
                        m=m+2*x+3;
                }
```

```
            else{
                m=m+2*(x-y)+5;
                y=y-1;
            }
            x=x+1;
            putpixel(x+cx,y+cy,2);
            putpixel(x+cx,-y+cy,2);
            putpixel(-x+cx,-y+cy,2);
            putpixel(-x+cx,y+cy,2);
            putpixel(y+cy,x+cx,2);
            putpixel(y+cy,-x+cx,2);
            putpixel(-y+cy,x+cx,2);
            putpixel(-y+cy,-x+cx,2);

        }
        getch();
        closegraph();

}
```

Midpoint Ellipse

```
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
int main()
{
int gd=DETECT, gm, p1, q1, angle1, angle2, xr, yr;
initgraph(&gd, &gm,(char*)"");
printf("Enter the centre coordinate(p1, q1):");
scanf("%d%d", &p1, &q1);
printf("Enter the semi-major axis:");
scanf("%d", &xr);
printf("Enter the semi-minor axis:");
scanf("%d", &yr);
angle1=0;
angle2=360;
setcolor(BROWN);
ellipse(p1,q1,angle1,angle2,xr,yr);
getch();
closegraph();
}
```

Koch curve

```
#include<stdio.h>
#include<graphics.h>
#include<math.h>

void koch(int x1,int y1,int x2,int y2,int it)
{
        float angle=60*M_PI/180;

        int x3=(2*x1+x2)/3;
        int y3=(2*y1+y2)/3;
        int x4=(x1+2*x2)/3;
        int y4=(y1+2*y2)/3;

        int x=x3+(x4-x3)*cos(angle)+(y4-y3)*sin(angle);
        int y=y3-(x4-x3)*sin(angle)+(y4-y3)*cos(angle);

        if(it>0){
        koch(x1,y1,x3,y3,it-1);
        koch(x3,y3,x,y,it-1);
        koch(x,y,x4,y4,it-1);
        koch(x4,y4,x2,y2,it-1);
        }
        else{
```

```
        line(x1,y1,x3,y3);
        line(x3,y3,x,y);
        line(x,y,x4,y4);
        line(x4,y4,x2,y2);
        }
}

int main(void)
{
        int gd=DETECT,gm;
        initgraph(&gd,&gm,(char*)"");
        int x1=100,y1=100,x2=400,y2=400;
        koch(x1,y1,x2,y2,8);
        getch();
        return 0;
}
```

Bitmap

```
#include<stdio.h>
#include<conio.h>
#include<process.h>
#include<math.h>
#include<graphics.h>

struct header
{
  int signature;
  long size;
  int resv1,resv2;
  long offset,BITMAP,width,height;
  int planes,bits;
  long compress,size_image,hres,vres,nocolour,impcolour;
} head;

  int main()
  {
        int i;
        FILE *fp;
        fp=fopen("c:\\TC\\untitled.bmp","rb+");

// r+ for read and write,b for binary //
```

```
//clrscr();
    if(fp==NULL)
    { printf("\n cannot open the file");
      getch();
      exit(0);
    }
    fread(&head,sizeof(head),1,fp);
    printf("\n attributes of image saved");
    printf("\n size of file in bytes=%d",head.size);
    printf("\n width=%d",head.width);
    printf("\n height=%d",head.height);
    printf("\n no of planes=%d",head.planes);
    printf("\n no of bits=%d",head.bits);
    printf("\n compression type=%d",head.compress);
    printf("\n size of image=%d",head.size_image);

    getch();
    }
```

Cohen Sutherland line clipping

```
#include<graphics.h>
#include<conio.h>
#include<stdio.h>
#include<math.h>
int main()
{
int rcode_begin[4]={0,0,0,0},rcode_end[4]={0,0,0,0},region_code[4];
int W_xmax,W_ymax,W_xmin,W_ymin,flag=0;
float slope;
int x,y,x1,y1,i, xc,yc;
int gr=DETECT,gm;
initgraph(&gr,&gm,"C:\\TURBOC3\\BGI");
printf("\n****** Cohen Sutherlsnd Line Clipping algorithm **********");
printf("\n Now, enter XMin, YMin =");

scanf("%d %d",&W_xmin,&W_ymin);
printf("\n First enter XMax, YMax =");
scanf("%d %d",&W_xmax,&W_ymax);
printf("\n Please enter intial point x and y= ");
scanf("%d %d",&x,&y);
printf("\n Now, enter final point x1 and y1= ");
scanf("%d %d",&x1,&y1);
cleardevice();
```

```
rectangle(W_xmin,W_ymin,W_xmax,W_ymax);
line(x,y,x1,y1);
line(0,0,600,0);
line(0,0,0,600);
if(y>W_ymax)  {
rcode_begin[0]=1;    // Top
flag=1 ;
}
if(y<W_ymin) {
rcode_begin[1]=1;         // Bottom
flag=1;
}
if(x>W_xmax)  {
rcode_begin[2]=1;         // Right
flag=1;
}
if(x<W_xmin)   {
rcode_begin[3]=1;         //Left
flag=1;
}

//end point of Line
if(y1>W_ymax){
rcode_end[0]=1;         // Top
flag=1;
}
```

```
if(y1<W_ymin) {
rcode_end[1]=1;          // Bottom
flag=1;
}
if(x1>W_xmax){
rcode_end[2]=1;          // Right
flag=1;
}
if(x1<W_xmin){
rcode_end[3]=1;          //Left
flag=1;
 }
if(flag==0)
{
printf("No need of clipping as it is already in window");
}
flag=1;
for(i=0;i<4;i++){
region_code[i]= rcode_begin[i] && rcode_end[i] ;
if(region_code[i]==1)
 flag=0;
}
if(flag==0)
{
printf("\n Line is completely outside the window");
}
```

```
else{
slope=(float)(y1-y)/(x1-x);
if(rcode_begin[2]==0 && rcode_begin[3]==1)   //left
{
y=y+(float) (W_xmin-x)*slope ;
x=W_xmin;

}
if(rcode_begin[2]==1 && rcode_begin[3]==0)      // right
{
y=y+(float) (W_xmax-x)*slope ;
x=W_xmax;

}
if(rcode_begin[0]==1 && rcode_begin[1]==0)     // top
{
x=x+(float) (W_ymax-y)/slope ;
y=W_ymax;

}
if(rcode_begin[0]==0 && rcode_begin[1]==1)    // bottom
{
x=x+(float) (W_ymin-y)/slope ;
y=W_ymin;

}
```

```
// end points
if(rcode_end[2]==0 && rcode_end[3]==1)   //left
{
y1=y1+(float) (W_xmin-x1)*slope ;
x1=W_xmin;

}
if(rcode_end[2]==1 && rcode_end[3]==0)      // right
{
y1=y1+(float) (W_xmax-x1)*slope ;
x1=W_xmax;

}
if(rcode_end[0]==1 && rcode_end[1]==0)     // top
{
x1=x1+(float) (W_ymax-y1)/slope ;
y1=W_ymax;

}
if(rcode_end[0]==0 && rcode_end[1]==1)     // bottom
{
x1=x1+(float) (W_ymin-y1)/slope ;
y1=W_ymin;

}
}
```

```
delay(1000);
clearviewport();
rectangle(W_xmin,W_ymin,W_xmax,W_ymax);
line(0,0,600,0);
line(0,0,0,600);
setcolor(RED);
line(x,y,x1,y1);
getch();
closegraph();
}
```

Sutherland Hodgman polygon clipping

```
#include<stdio.h>
#include<graphics.h>
#include<conio.h>
#include<stdlib.h>
int main()
```

```
{
int gd,gm,n,*x,i,k=0;
int w[]={220,140,420,140,420,340,220,340,220,140}; //array for drawing window
detectgraph(&gd,&gm);
initgraph(&gd,&gm,(char*)"");
printf("Window:-");
setcolor(RED);
drawpoly(5,w); //window drawn
printf("Enter the no. of vertices of polygon: ");
scanf("%d",&n);
x = (int*)malloc(n*2+1);
printf("Enter the coordinates of points:\n");
k=0;
for(i=0;i<n*2;i+=2) //reading vertices of polygon
{
printf("(x%d,y%d): ",k,k);
scanf("%d,%d",&x[i],&x[i+1]);
k++;
}
x[n*2]=x[0]; //assigning the coordinates of first vertex to last additional vertex for drawpoly method.
x[n*2+1]=x[1];
setcolor(WHITE);
drawpoly(n+1,x);
printf("\nPress a button to clip a polygon..");
getch();
```

```
setcolor(RED);
drawpoly(5,w);
setfillstyle(SOLID_FILL,BLACK);
floodfill(2,2,RED);
printf("\nThis is the clipped polygon..");
getch();
cleardevice();
closegraph();
return 0;
}
```