**PROGRAM 10**

**Write a program to draw a circular arc.**

#include<stdio.h>

#include<graphics.h>

#include<math.h>

#define ROUND(a) ((int)(a+0.5))

void arccircle(int xc,int yc, int r,float a1,float a2)

{

int x = ROUND(r\*cos(a2));

int y = ROUND(r\*sin(a2));

int xf = ROUND(r\*cos(a1));

int yf = ROUND(r\*sin(a1));

int p=5/4-r;

putpixel(xc+x,yc+y,15);

while(x<=xf && y>=yf)

{

if(x<y)

{

x++;

if(p<0)

p+= 2\*x+1;

else

{

y--;

p+=2\*(x-y)+1;

}

putpixel(xc+x,yc+y,15);

}

else

{

y--;

if(p>0)

p-= 2\*y +1;

else

{

x++;

p-=2\*(y-x)+1;

}

putpixel(xc+x,yc+y,15);

}

}

}

int main()

{

int xc,yc,r;

float a1,a2;

int gdriver = DETECT, gmode, errorcode;

initgraph(&gdriver, &gmode, "..\\");

errorcode = graphresult();

if (errorcode != grOk)

{

printf("Graphics error: %s\n", grapherrormsg(errorcode));

printf("Press any key to halt:");

getch();

exit(1);

}

printf("Enter center of circle\n");

scanf("%d %d", &xc, &yc);

printf("Enter radius\n");

scanf("%d", &r);

printf("Enter the start and end angles with x-axis\n");

scanf("%f%f",&a1,&a2);

a1=a1\*3.142/180;

a2=a2\*3.142/180;

arccircle(xc,yc,r,a1,a2);

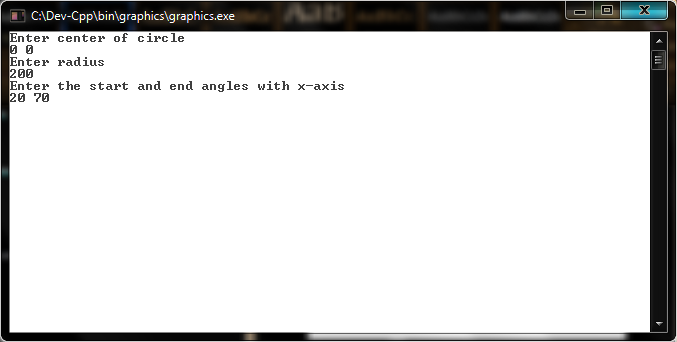
getch();

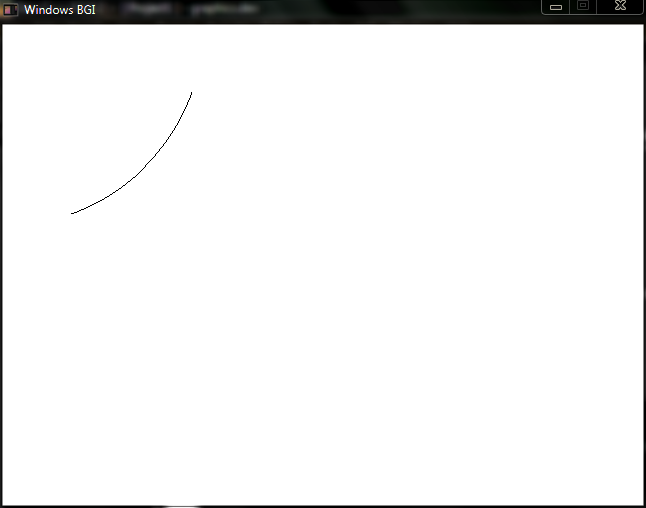
closegraph();

return 0;

}

**OUTPUT 10**





**PROGRAM 11**

**Write a program to draw an elliptical arc.**

#include<stdio.h>

#include<graphics.h>

#include<math.h>

#define ROUND(a) ((int)(a+0.5))

void arcellipse(int xc, int yc, int Rx, int Ry,float a1,float a2)

{

int Rx2= Rx\*Rx;

int Ry2=Ry\*Ry;

int twoRx2= 2\*Rx2;

int twoRy2= 2\*Ry2;

int p;

int x=ROUND(Rx\*cos(a2));

int y=ROUND(Ry\*sin(a2));

int xf=ROUND(Rx\*cos(a1));

int yf=ROUND(Ry\*sin(a1));

int px= twoRy2\*x;

int py= twoRx2\*y;

putpixel(xc+x,yc+y,15);

p=ROUND(Ry2-(Rx2\*Ry)+(0.25\*Rx2));

while(px<py)

{

x++;

px+=twoRy2;

if(p<0)

p+= Ry2+px;

else

{

y--;

py-=twoRx2;

p+=Ry2-py+px;

}

putpixel(xc+x,yc+y,WHITE);

}

p= ROUND(Ry2\*(x+0.5)\*(x+0.5)+Rx2\*(y-1)\*(y-1)-Rx2\*Ry2);

while(y>yf)

{

y--;

py-=twoRx2;

if(p>0)

p+=Rx2-py;

else

{

x++;

px+=twoRy2;

p+=Rx2-py+px;

}

putpixel(xc+x,yc+y,WHITE);

}

}

int main()

{

int xc,yc, Rx, Ry;

float a1,a2;

int gdriver = DETECT, gmode, errorcode;

initgraph(&gdriver, &gmode, "..\\");

errorcode = graphresult();

if (errorcode != grOk)

{

printf("Graphics error: %s\n", grapherrormsg(errorcode));

printf("Press any key to halt:");

getch();

exit(1);

}

printf("Enter center of ellipse\n");

scanf("%d %d", &xc, &yc);

printf("Enter the x-radius and y-radius\n");

scanf("%d %d",&Rx,&Ry);

printf("Enter the start and end angles with x-axis\n");

scanf("%f %f",&a1,&a2);

a1=a1\*3.142/180;

a2=a2\*3.142/180;

arcellipse(xc,yc,Rx,Ry,a1,a2);

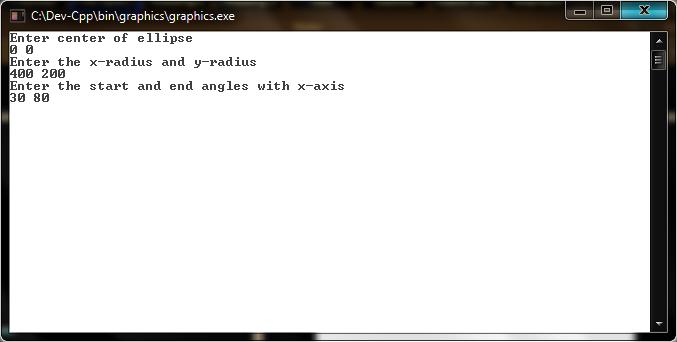
getch();

closegraph();

return 0;

}

**OUTPUT 11**





**PROGRAM 12**

**Write a program to implement Flood Fill algorithm.**

#include<stdio.h>

#include<graphics.h>

void floodFill(int x,int y)

{

int current;

int old=0;

current = getpixel(x,y);

if(current==old)

{

putpixel(x,y,15);

floodFill(x+1,y);

floodFill(x-1,y);

floodFill(x,y+1);

floodFill(x,y-1);

}

else return;

}

void setpixel(int xc, int yc, int x, int y)

{

putpixel(xc+x,yc+y,15);

putpixel(xc+x,yc-y,15);

putpixel(xc-x,yc+y,15);

putpixel(xc-x,yc-y,15);

putpixel(xc+y,yc+x,15);

putpixel(xc+y,yc-x,15);

putpixel(xc-y,yc+x,15);

putpixel(xc-y,yc-x,15);

}

void midptcircle(int xc, int yc, int r)

{

int p = 1 - r;

int x= 0, y= r;

setpixel(xc,yc,x,y);

while(x<y)

{

x++;

if(p<0)

{

p+= 2\*x +1;

}

else

{

y--;

p+= 2\*(x-y) +1;

}

setpixel(xc,yc,x,y);

}

}

int main()

{

int xc, yc, r;

int gdriver = DETECT, gmode, errorcode;

initgraph(&gdriver, &gmode, "..\\");

errorcode = graphresult();

if (errorcode != grOk)

{

printf("Graphics error: %s\n", grapherrormsg(errorcode));

printf("Press any key to halt:");

getch();

exit(1);

}

printf("Enter center of circle\n");

scanf("%d %d", &xc, &yc);

printf("Enter radius\n");

scanf("%d", &r);

midptcircle(xc,yc,r);

floodFill(xc,yc);

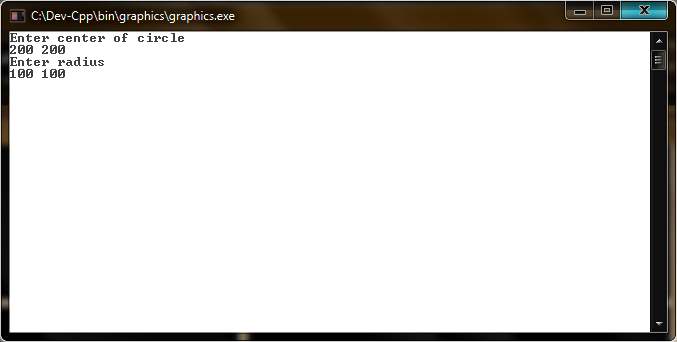
getch();

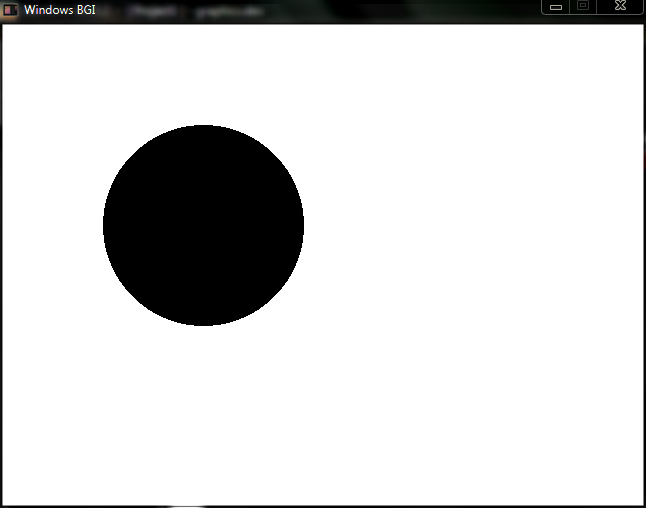
closegraph();

return 0;

}

**OUTPUT 12**





**PROGRAM 13**

**Write a program to implement Boundary Fill algorithm.**

#include<stdio.h>

#include<graphics.h>

void boundaryFill(int x, int y, int f, int b)

{

if((getpixel(x,y)!=b)&&(getpixel(x,y)!=f))

{

putpixel(x,y,f);

boundaryFill(x+1,y,f,b);

boundaryFill(x-1,y,f,b);

boundaryFill(x,y+1,f,b);

boundaryFill(x,y-1,f,b);

}

}

void setpixel(int xc, int yc, int x, int y)

{

putpixel(xc+x,yc+y,15);

putpixel(xc+x,yc-y,15);

putpixel(xc-x,yc+y,15);

putpixel(xc-x,yc-y,15);

putpixel(xc+y,yc+x,15);

putpixel(xc+y,yc-x,15);

putpixel(xc-y,yc+x,15);

putpixel(xc-y,yc-x,15);

}

void midptcircle(int xc, int yc, int r)

{

int p = 1 - r;

int x= 0, y= r;

setpixel(xc,yc,x,y);

while(x<y)

{

x++;

if(p<0)

{

p+= 2\*x +1;

}

else

{

y--;

p+= 2\*(x-y) +1;

}

setpixel(xc,yc,x,y);

}

}

int main()

{

int xc, yc, r;

int gdriver = DETECT, gmode, errorcode;

initgraph(&gdriver, &gmode, "..\\");

errorcode = graphresult();

if (errorcode != grOk)

{

printf("Graphics error: %s\n", grapherrormsg(errorcode));

printf("Press any key to halt:");

getch();

exit(1);

}

printf("Enter center of circle\n");

scanf("%d %d", &xc, &yc);

printf("Enter radius\n");

scanf("%d", &r);

midptcircle(xc,yc,r);

boundaryFill(xc,yc,15,15);

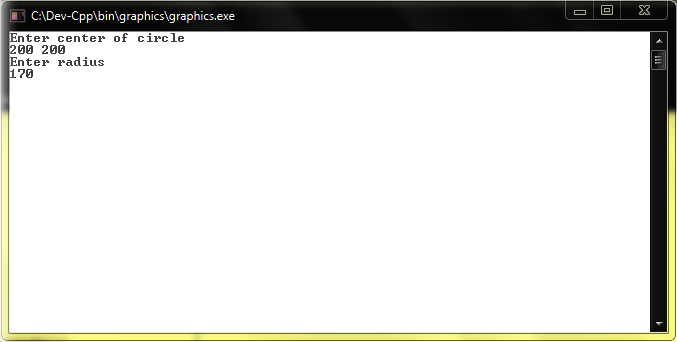
getch();

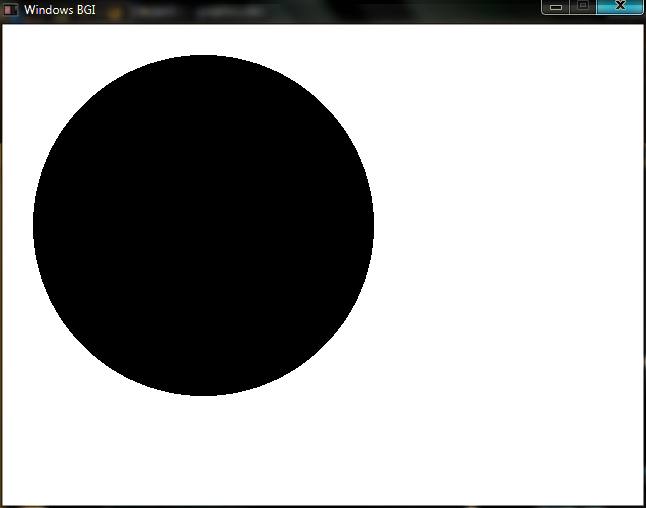
closegraph();

return 0;

}

**OUTPUT 13**





**PROGRAM 14**

**Write a program to implement 2-D translation.**

#include<stdio.h>

#include<graphics.h>

#define ROUND(a) ((int)(a+0.5))

void ddaline(int x1, int y1, int x2, int y2)

{

float xsteps, ysteps, x=x1, y=y1;

int dx = x2-x1;

int dy = y2-y1;

int steps,k=1;

if(abs(dx)>=abs(dy))

steps=abs(dx);

else steps=abs(dy);

xsteps= dx/(float)steps;

ysteps= dy/(float)steps;

putpixel(ROUND(x),ROUND(y),15);

while(k<=steps)

{

x+=xsteps;

y+=ysteps;

putpixel(ROUND(x), ROUND(y),15);

k++;

}

}

void translate(int x1, int y1, int x2, int y2, int tx, int ty)

{

ddaline(x1,y1,x2,y2);

ddaline(x1+tx,y1+ty,x2+tx,y2+ty);

}

int main()

{

int x1, x2, y1, y2, tx, ty;

int gdriver = DETECT, gmode, errorcode;

initgraph(&gdriver, &gmode, "..\\");

errorcode = graphresult();

if (errorcode != grOk)

{

printf("Graphics error: %s\n", grapherrormsg(errorcode));

printf("Press any key to halt:");

getch();

exit(1);

}

printf("Enter start point\n");

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

printf("Enter end point\n");

scanf("%d %d", &x2, &y2);

printf("Enter value for \'tx\' and \'ty\' \n");

scanf("%d %d", &tx, &ty);

translate(x1, y1, x2, y2, tx, ty);

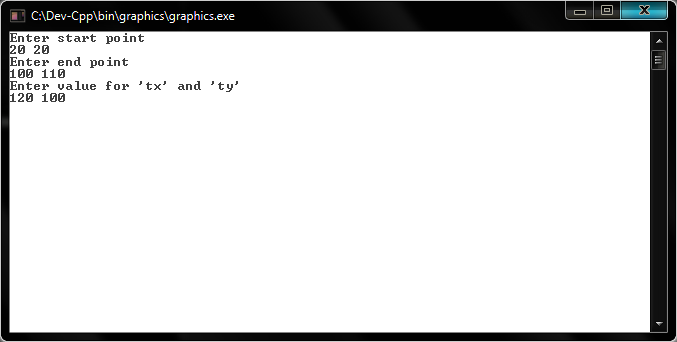
getch();

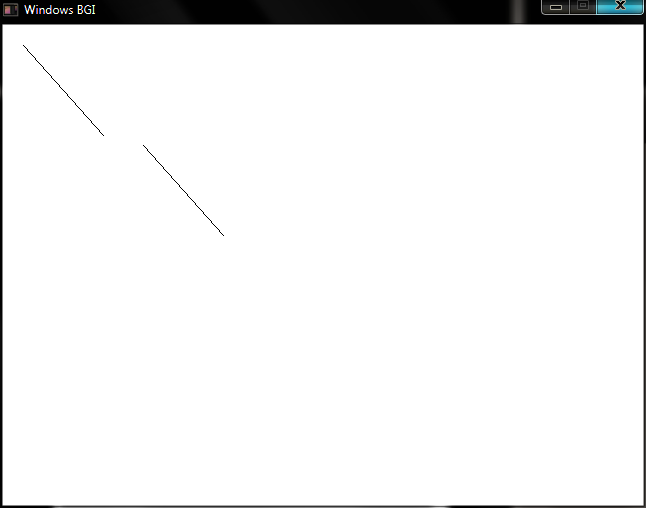
closegraph();

return 0;

}

**OUTPUT 14**





**PROGRAM 15**

**Write a program to implement 2-D rotation.**

#include<stdio.h>

#include<graphics.h>

#include<math.h>

#define ROUND(a) ((int)(a+0.5))

void ddaline(int x1, int y1, int x2, int y2)

{

float xsteps, ysteps, x=x1, y=y1;

int dx = x2-x1;

int dy = y2-y1;

int steps,k=1;

if(abs(dx)>=abs(dy))

steps=abs(dx);

else steps=abs(dy);

xsteps= dx/(float)steps;

ysteps= dy/(float)steps;

putpixel(ROUND(x),ROUND(y),15);

while(k<=steps)

{

x+=xsteps;

y+=ysteps;

putpixel(ROUND(x), ROUND(y),15);

k++;

}

}

void rotate(int x1, int y1, int x2, int y2, float theta)

{

int xtmp, ytmp;

xtmp = x1 + ROUND ((x2-x1)\*cos(theta) - (y2-y1)\*sin(theta));

ytmp = y1 + ROUND ((x2-x1)\*sin(theta) + (y2-y1)\*cos(theta));

ddaline(x1,y1,x2,y2);

ddaline(x1,y1,xtmp,ytmp);

}

int main()

{

int x1, x2, y1, y2;

float theta;

int gdriver = DETECT, gmode, errorcode;

initgraph(&gdriver, &gmode, "..\\");

errorcode = graphresult();

if (errorcode != grOk)

{

printf("Graphics error: %s\n", grapherrormsg(errorcode));

printf("Press any key to halt:");

getch();

exit(1);

}

printf("Enter start point\n");

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

printf("Enter end point\n");

scanf("%d %d", &x2, &y2);

printf("Enter value of angle to rotate line about starting point\n");

scanf("%f", &theta);

rotate(x1, y1, x2, y2, theta);

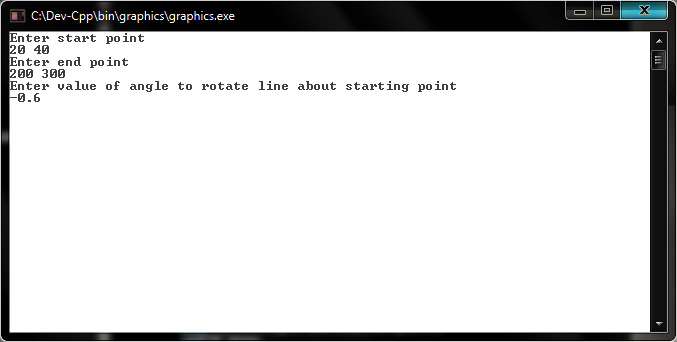
getch();

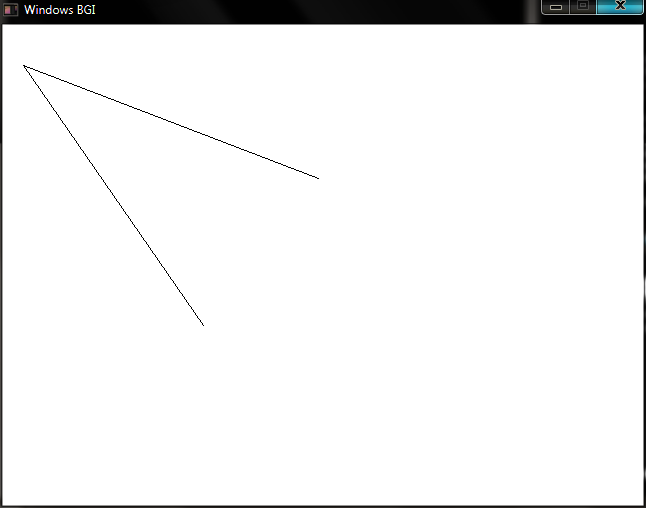
closegraph();

return 0;

}

**OUTPUT 15**





**PROGRAM 16**

**Write a program to implement 2-D reflection.**

#include<stdio.h>

#include<graphics.h>

#define ROUND(a) ((int)(a+0.5))

void ddaline(int x1, int y1, int x2, int y2)

{

float xsteps, ysteps, x=x1, y=y1;

int dx = x2-x1;

int dy = y2-y1;

int steps,k=1;

if(abs(dx)>=abs(dy))

steps=abs(dx);

else steps=abs(dy);

xsteps= dx/(float)steps;

ysteps= dy/(float)steps;

putpixel(ROUND(x),ROUND(y),15);

while(k<=steps)

{

x+=xsteps;

y+=ysteps;

putpixel(ROUND(x), ROUND(y),15);

k++;

}

}

void reflect(int x1, int y1, int x2, int y2, int Y)

{

ddaline(x1, y1, x2, y2);

ddaline(0, Y, 400, Y);

ddaline(x1, 2\*Y-y1, x2, 2\*Y-y2);

}

int main()

{

int x1, x2, y1, y2, Y;

int gdriver = DETECT, gmode, errorcode;

initgraph(&gdriver, &gmode, "..\\");

errorcode = graphresult();

if (errorcode != grOk)

{

printf("Graphics error: %s\n", grapherrormsg(errorcode));

printf("Press any key to halt:");

getch();

exit(1);

}

printf("Enter start point\n");

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

printf("Enter end point\n");

scanf("%d %d", &x2, &y2);

printf("Enter value for \'y\' about which line is to be reflected\n");

scanf("%d", &Y);

reflect(x1, y1, x2, y2, Y);

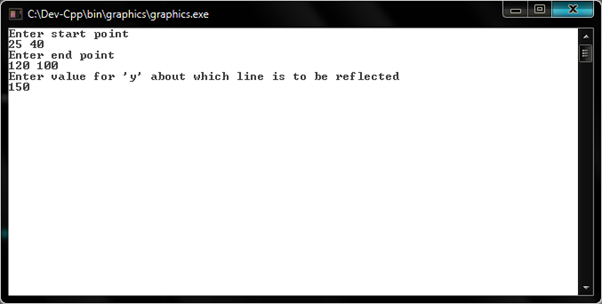
getch();

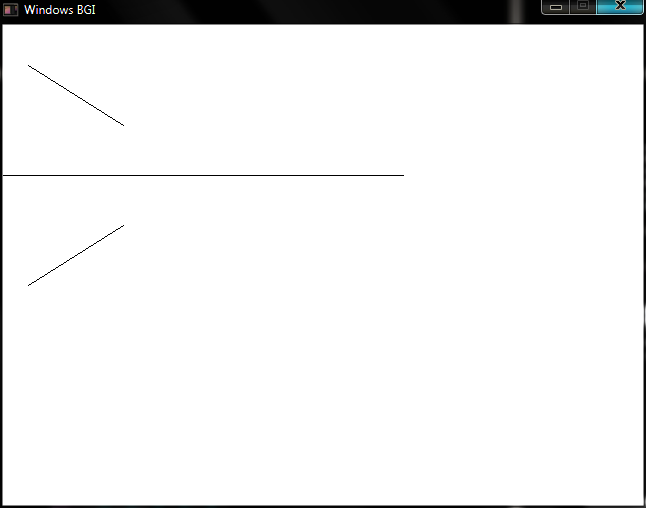
closegraph();

return 0;

}

**OUTPUT 16**





**PROGRAM 17**

**Write a program to implement 2-D scaling.**

#include<stdio.h>

#include<graphics.h>

#define ROUND(a) ((int)(a+0.5))

void ddaline(int x1, int y1, int x2, int y2)

{

float xsteps, ysteps, x=x1, y=y1;

int dx = x2-x1;

int dy = y2-y1;

int steps,k=1;

if(abs(dx)>=abs(dy))

steps=abs(dx);

else steps=abs(dy);

xsteps= dx/(float)steps;

ysteps= dy/(float)steps;

putpixel(ROUND(x),ROUND(y),15);

while(k<=steps)

{

x+=xsteps;

y+=ysteps;

putpixel(ROUND(x), ROUND(y),15);

k++;

}

}

void scale(int x1, int y1, int x2, int y2, int Sx, int Sy)

{

ddaline(x1,y1,x2,y2);

ddaline(Sx\*x1,Sy\*y1,Sx\*x2,Sy\*y2);

}

int main()

{

int x1, x2, y1, y2, Sx, Sy;

int gdriver = DETECT, gmode, errorcode;

initgraph(&gdriver, &gmode, "..\\");

errorcode = graphresult();

if (errorcode != grOk)

{

printf("Graphics error: %s\n", grapherrormsg(errorcode));

printf("Press any key to halt:");

getch();

exit(1);

}

printf("Enter start point\n");

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

printf("Enter end point\n");

scanf("%d %d", &x2, &y2);

printf("Enter value for \'Sx\' and \'Sy\' \n");

scanf("%d %d", &Sx, &Sy);

scale(x1, y1, x2, y2, Sx, Sy);

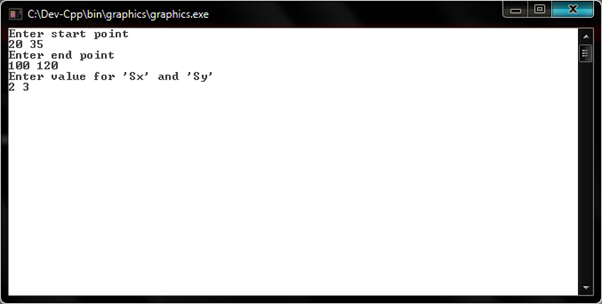
getch();

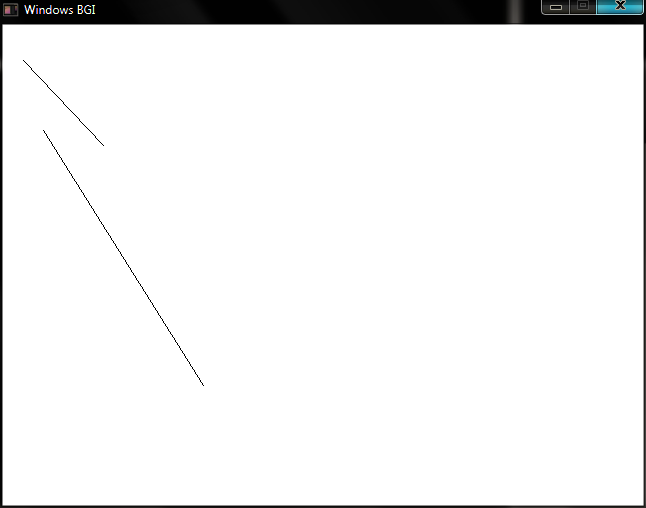
closegraph();

return 0;

}

**OUTPUT 17**





**PROGRAM 18**

**Write a program to implement 2-D shearing.**

#include<stdio.h>

#include<graphics.h>

#define ROUND(a) ((int)(a+0.5))

void ddaline(int x1, int y1, int x2, int y2)

{

float xsteps, ysteps, x=x1, y=y1;

int dx = x2-x1;

int dy = y2-y1;

int steps,k=1;

if(abs(dx)>=abs(dy))

steps=abs(dx);

else steps=abs(dy);

xsteps= dx/(float)steps;

ysteps= dy/(float)steps;

putpixel(ROUND(x),ROUND(y),15);

while(k<=steps)

{

x+=xsteps;

y+=ysteps;

putpixel(ROUND(x), ROUND(y),15);

k++;

}

}

void quadilateral(int x[4], int y[4])

{

int i=0;

while(i<4)

{

if(i==3)

{

ddaline(x[i],y[i],x[0],y[0]);

return;

}

ddaline (x[i],y[i],x[i+1],y[i+1]);

i++;

}

}

void shear(int x[4], int y[4], float sh, int Y)

{

int shx[4],i=0;

while(i<4)

{

shx[i]=x[i] + ROUND(sh\*(y[i]-Y));

i++;

}

quadilateral(x,y);

ddaline(0,Y,400,Y);

quadilateral(shx,y);

}

int main()

{

int x[4],y[4],i=0, Y;

float sh;

int gdriver = DETECT, gmode, errorcode;

initgraph(&gdriver, &gmode, "..\\");

errorcode = graphresult();

if (errorcode != grOk)

{

printf("Graphics error: %s\n", grapherrormsg(errorcode));

printf("Press any key to halt:");

getch();

exit(1);

}

printf("Enter quadilateral points in sequence\n");

while(i<4)

{

scanf("%d %d", &x[i], &y[i]);

i++;

}

printf("Enter value for shear factor and \'y\' reference\n");

scanf("%f %d", &sh, &Y);

shear(x,y,sh,Y);

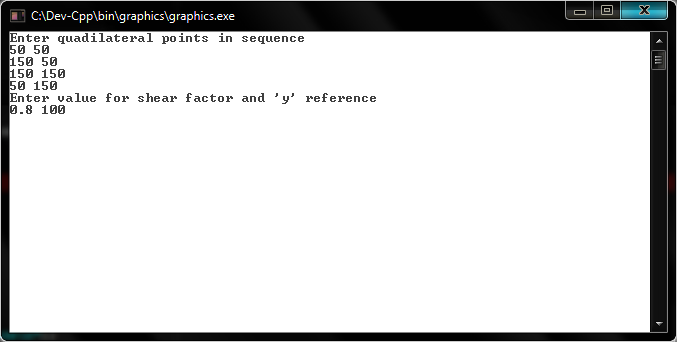
getch();

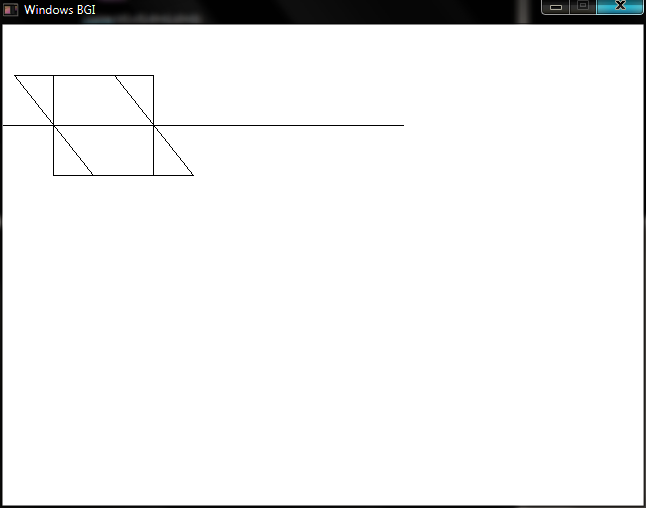
closegraph();

return 0;

}

**OUTPUT 18**





**PROGRAM 19**

**Write a program to implement Cohen Sutherland Algorithm.**

#include<graphics.h>

#include<stdio.h>

#define TOP 0x1

#define BOTTOM 0x2

#define RIGHT 0x4

#define LEFT 0x8

#define ROUND(a) ((int)(a+0.5))

void ddaline(int x1, int y1, int x2, int y2)

{

float xsteps, ysteps, x=x1, y=y1;

int dx = x2-x1;

int dy = y2-y1;

int steps,k=1;

if(abs(dx)>=abs(dy))

steps=abs(dx);

else steps=abs(dy);

xsteps= dx/(float)steps;

ysteps= dy/(float)steps;

putpixel(ROUND(x),ROUND(y),15);

while(k<=steps)

{

x+=xsteps;

y+=ysteps;

putpixel(ROUND(x), ROUND(y),15);

k++;

}

}

int calcode (float x,float y,float xwmin, float ywmin,float xwmax,float ywmax)

{

int code =0;

if(y> ywmax)

code |=TOP;

else if( y<ywmin)

code |= BOTTOM;

else if(x > xwmax)

code |= RIGHT;

else if ( x< xwmin)

code |= LEFT;

return(code);

}

void lineclip(float x0,float y0,float x1,float y1,float xwmin,float ywmin,float xwmax,float ywmax )

{

unsigned int code0,code1,codeout;

int accept = 0, done=0;

code0 = calcode(x0,y0,xwmin,ywmin,xwmax,ywmax);

code1 = calcode(x1,y1,xwmin,ywmin,xwmax,ywmax);

do{

if(!(code0 | code1))

{ accept =1 ; done =1; }

else

if(code0 & code1) done = 1;

else

{

float x,y;

codeout = code0 ? code0 : code1;

if(codeout & TOP)

{

x = x0 + (x1-x0)\*(ywmax-y0)/(y1-y0);

y = ywmax;

}

else

if( codeout & BOTTOM)

{

x = x0 + (x1-x0)\*(ywmin-y0)/(y1-y0);

y = ywmin;

}

else

if ( codeout & RIGHT)

{

y = y0+(y1-y0)\*(xwmax-x0)/(x1-x0);

x = xwmax;

}

else

{

y = y0 + (y1-y0)\*(xwmin-x0)/(x1-x0);

x = xwmin;

}

if( codeout == code0)

{

x0 = x; y0 = y;

code0=calcode(x0,y0,xwmin,ywmin,xwmax,ywmax);

}

else

{

x1 = x; y1 = y;

code1 = calcode(x1,y1,xwmin,ywmin,xwmax,ywmax);

}

}

} while( done == 0);

if(accept) ddaline(x0,y0,x1,y1);

ddaline(xwmin,ywmin,xwmin,ywmax);

ddaline(xwmin,ywmax,xwmax,ywmax);

ddaline(xwmax,ywmax,xwmax,ywmin);

ddaline(xwmax,ywmin,xwmin,ywmin);

getch();

}

int main()

{

float x2,y2,x1,y1,xwmin,ywmin,xwmax,ywmax;

int gdriver = DETECT, gmode, errorcode;

initgraph(&gdriver, &gmode, "..\\");

errorcode = graphresult();

if (errorcode != grOk)

{

printf("Graphics error: %s\n", grapherrormsg(errorcode));

printf("Press any key to halt:");

getch();

exit(1);

}

printf("Enter the starting point\n");

scanf("%f %f", &x1, &y1);

printf("Enter the ending point\n");

scanf("%f %f", &x2, &y2);

printf("Enter xwmin, ywmin, xwmax, ywmax\n");

scanf("%f %f %f %f",&xwmin,&ywmin,&xwmax,&ywmax);

lineclip(x1,y1,x2,y2,xwmin,ywmin,xwmax,ywmax );

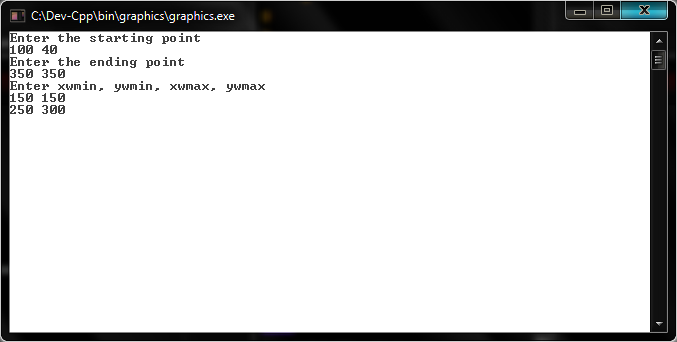
getch();

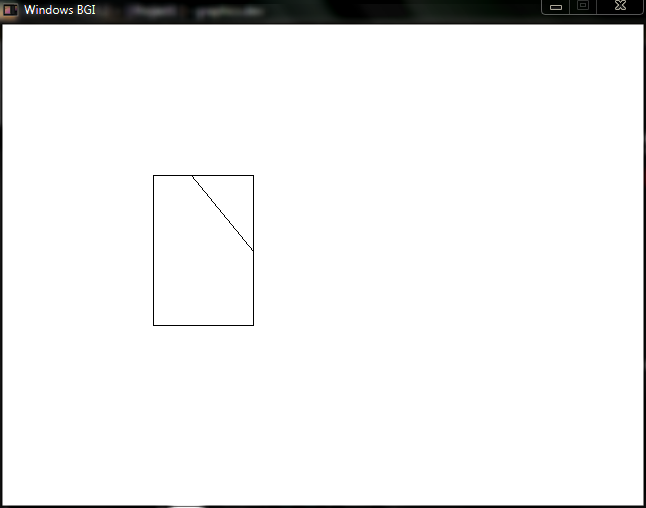
closegraph();

return 0;

}

**OUTPUT 19**





**PROGRAM 20**

**Write a program to implement Liang Barsky Algorithm.**

#include<stdio.h>

#include<graphics.h>

#define ROUND(a) ((int)(a+0.5))

void ddaline(int x1, int y1, int x2, int y2)

{

float xsteps, ysteps, x=x1, y=y1;

int dx = x2-x1;

int dy = y2-y1;

int steps,k=1;

if(abs(dx)>=abs(dy))

steps=abs(dx);

else steps=abs(dy);

xsteps= dx/(float)steps;

ysteps= dy/(float)steps;

putpixel(ROUND(x),ROUND(y),15);

while(k<=steps)

{

x+=xsteps;

y+=ysteps;

putpixel(ROUND(x), ROUND(y),15);

k++;

}

}

void lineclip(float xwmin,float ywmin,float xwmax,float ywmax,float x1,float y1,float x2,float y2)

{

float x11,y11,x21,y21;

float xdelta=x2-x1;

float ydelta=y2-y1;

float p=0,q=0,r=0;

float t0=0;

float t1=1;

for(int edge =0;edge<4;edge++)

{

if(edge==0)

{

p=-xdelta;

q=-(xwmin-x1);

}

else if(edge==1)

{

p=xdelta;

q=(xwmax-x1);

}

else if(edge==2)

{

p=-ydelta;

q=-(ywmin-y1);

}

else if(edge==3)

{

p=ydelta;

q=(ywmax-y1);

}

r=q/p;

if(p==0 && q<0)

{

printf("The line is totaly outside the clipping window.\n");

return ;

}

if(p<0)

{

if(r>t1)

{

printf("The line is totaly outside the clipping window.\n");

return ;

}

else if(r>t0)

t0=r;

}

else if(p>0)

{

if(r<t0)

{

printf("The line is totaly outside the clipping window.\n");

return ;

}

else if(r<t1)

t1=r;

}

}

if(t0>0)

{ x11=x1+t0\*xdelta;

y11=y1+t0\*ydelta;

}

if(t1<1)

{

x21=x1+t1\*xdelta;

y21=y1+t1\*ydelta;

}

ddaline(x11,y11,x21,y21);

ddaline(xwmin,ywmin,xwmin,ywmax);

ddaline(xwmin,ywmax,xwmax,ywmax);

ddaline(xwmax,ywmax,xwmax,ywmin);

ddaline(xwmax,ywmin,xwmin,ywmin);

}

int main()

{

float x2,y2,x1,y1,xwmin,ywmin,xwmax,ywmax;

int gdriver = DETECT, gmode, errorcode;

initgraph(&gdriver, &gmode, "..\\");

errorcode = graphresult();

if (errorcode != grOk)

{

printf("Graphics error: %s\n", grapherrormsg(errorcode));

printf("Press any key to halt:");

getch();

exit(1);

}

printf("Enter the starting point\n");

scanf("%f %f", &x1, &y1);

printf("Enter the ending point\n");

scanf("%f %f", &x2, &y2);

printf("Enter xwmin, ywmin, xwmax, ywmax\n");

scanf("%f %f %f %f",&xwmin,&ywmin,&xwmax,&ywmax);

lineclip(xwmin,ywmin,xwmax,ywmax,x1,y1,x2,y2);

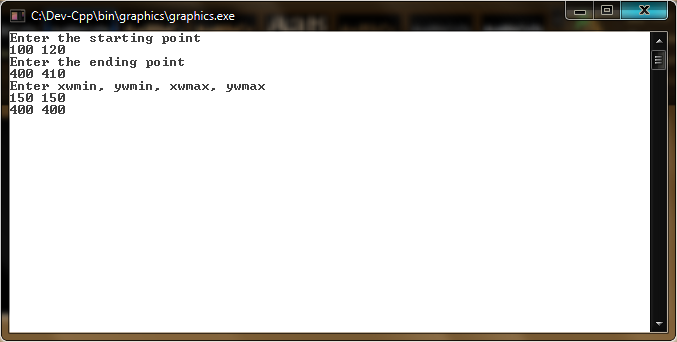
getch();

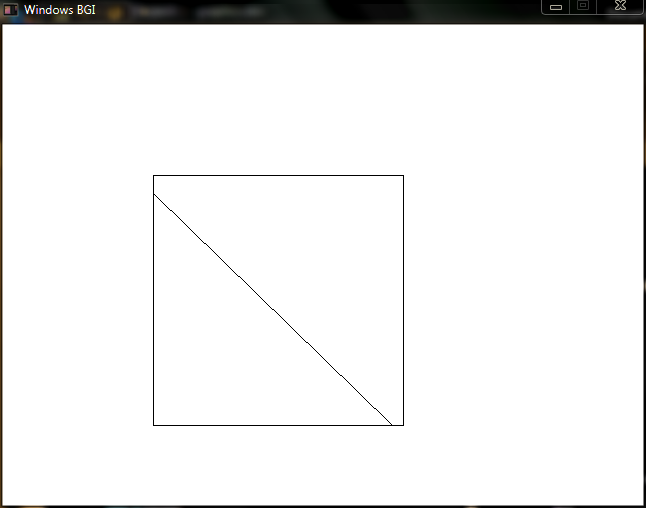
closegraph();

return 0;

}

**OUTPUT 20**





**PROGRAM 21**

**Write a program to implement 3-D perspective projection.**

#include<stdio.h>

#include<graphics.h>

#define ROUND(a) (int)(a+0.5)

void ddaline(int x1, int y1, int x2, int y2)

{

float xsteps, ysteps, x=x1, y=y1;

int dx = x2-x1;

int dy = y2-y1;

int steps,k=1;

if(abs(dx)>=abs(dy))

steps=abs(dx);

else steps=abs(dy);

xsteps= dx/(float)steps;

ysteps= dy/(float)steps;

putpixel(ROUND(x),ROUND(y),15);

while(k<=steps)

{

x+=xsteps;

y+=ysteps;

putpixel(ROUND(x), ROUND(y),15);

k++;

}

}

void project3d(int x[8],int y[8],int z[8],float xPrp,float yPrp,float zPrp)

{

int xp[8],yp[8],i;

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

{

xp[i]=ROUND(x[i]\*(zPrp/(zPrp-z[i]))- xPrp\*(z[i]/(zPrp-z[i])));

yp[i]=ROUND(y[i]\*(zPrp/(zPrp-z[i]))- yPrp\*(z[i]/(zPrp-z[i])));

}

ddaline(xp[0],yp[0],xp[1],yp[1]);

ddaline(xp[1],yp[1],xp[2],yp[2]);

ddaline(xp[2],yp[2],xp[3],yp[3]);

ddaline(xp[3],yp[3],xp[0],yp[0]);

ddaline(xp[4],yp[4],xp[5],yp[5]);

ddaline(xp[5],yp[5],xp[6],yp[6]);

ddaline(xp[6],yp[6],xp[7],yp[7]);

ddaline(xp[7],yp[7],xp[4],yp[4]);

ddaline(xp[0],yp[0],xp[4],yp[4]);

ddaline(xp[1],yp[1],xp[5],yp[5]);

ddaline(xp[2],yp[2],xp[6],yp[6]);

ddaline(xp[3],yp[3],xp[7],yp[7]);

}

int main()

{

int x[8],y[8],z[8],i;

float xPrp,yPrp,zPrp;

int gdriver = DETECT, gmode, errorcode;

initgraph(&gdriver, &gmode, "..\\");

errorcode = graphresult();

if (errorcode != grOk)

{

printf("Graphics error: %s\n", grapherrormsg(errorcode));

printf("Press any key to halt:");

getch();

exit(1);

}

printf("Enter the cube coordinates in sequence\n");

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

scanf("%d %d %d",&x[i],&y[i],&z[i]);

printf("Enter the projection reference point(PRP)\n");

scanf("%f %f %f",&xPrp,&yPrp,&zPrp);

project3d(x,y,z,xPrp,yPrp,zPrp);

getch();

closegraph();

return 0;

}

**OUTPUT 21**

