# 1. WAP A PROGRAM TO IMPLEMENT DDA ALGORITHM

#include<stdio.h>

#include<conio.h>

#include<graphics.h>

#include<math.h>

int main() {

int x1, y1, x2, y2, i, dxn, dyn;

float x, y, dx, dy, length;

int gd = DETECT, gm;

printf("enter initial points:");

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

printf("enter final points:");

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

initgraph( & gd, & gm, "../BGI");

dx = (x2 - x1);

dy = (y2 - y1);

if (abs(x2 - x1) > abs(y2 - y1)) {

length = abs(x2 - x1);

} else {

length = abs(y2 - y1);

}

x = x1;

y = y1;

i = 0;

dxn = dx / length;

dyn = dy / length;

putpixel(x, y, 4);

for (i = 0; i < length; i++) {

x = x + dxn;

y = y + dyn;

putpixel(x, y, 4);

}

getch();

closegraph();

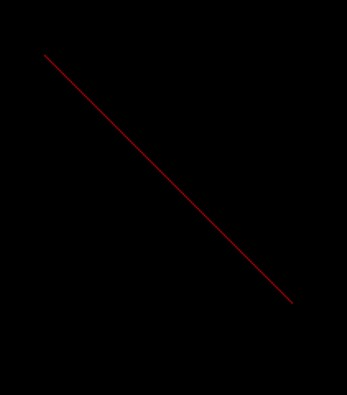
return 0;

}

OUTPUT:

enter initial points:100 100

enter final points:250 250



# 2. WRITE A PROGRAM TO IMPLEMENT BRESENHAM’S LINE DRAWING ALGORITHM

#include<stdio.h>

#include<conio.h>

#include<graphics.h>

#include<math.h>

int main() {

int x, y, x1, y1, x2, y2, dx, dy, p, i;

int gd = DETECT, gm;

printf("Enter initial points:");

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

printf("Enter final points:");

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

initgraph( & gd, & gm, "../BGI");

dx = (x2 - x1);

dy = (y2 - y1);

p = 2 \* dy - dx;

x = x1;

y = y1;

i = 0;

while (i <= dx) {

putpixel(x, y, GREEN);

if (p < 0) {

x = x + 1;

p = p + 2 \* dy;

} else {

x = x + 1;

y = y + 1;

p = p + 2 \* dy - 2 \* dx;

}

i++;

}

getch();

closegraph();

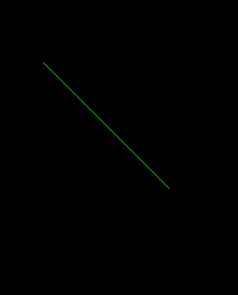
return 0;

}

OUTPUT:

Enter initial points:100 200

Enter final points:200 300



# 3. WRITE A PROGRAM TO IMPLEMENT MIDPOINT CIRCLE DRAWING ALGORITHM

#include<stdio.h>

#include<conio.h>

#include<graphics.h>

#include<math.h>

int main() {

int x, y, xc, yc, p, r;

clrscr();

int gd = DETECT, gm;

printf("Enter center of the circle:");

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

printf("Enter radius:");

scanf("%d", & r);

initgraph( & gd, & gm, "../BGI");

p = 1 - r;

x = 0;

y = r;

while (x <= y) {

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

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

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

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

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

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

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

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

if (p < 0) {

x = x + 1;

p = p + 2 \* x + 3;

} else {

x = x + 1;

y = y - 1;

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

}

}

getch();

closegraph();

return 0;

}

OUTPUT:

Enter center of the circle:50 50

Enter radius:20



# 4. WRITE A PROGRAM TO IMPLEMENT ELLIPSE DRAWING ALGORITHM

#include<stdio.h>

#include<conio.h>

#include<graphics.h>

#include<math.h>

void disp();

float x, y;

int xc, yc;

void main() {

int gd = DETECT, gm;

int rx, ry;

float p1, p2; clrscr();

initgraph( & gd, & gm, "C:\\TurboC3\\BGI");

printf("Enter the center point :");

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

printf("Enter the value for Rx and Ry :");

scanf("%d%d", & rx, & ry);

x = 0; y = ry;

disp();

p1 = (ry \* ry) - (rx \* rx \* ry) + (rx \* rx) / 4;

while ((2.0 \* ry \* ry \* x) <= (2.0 \* rx \* rx \* y)) {

x++;

if (p1 <= 0) {

p1 = p1 + (2.0 \* ry \* ry \* x) + (ry \* ry);

} else {

y--;

p1 = p1 + (2.0 \* ry \* ry \* x) - (2.0 \* rx \* rx \* y) + (ry \* ry);

}

disp();

x = -x;

disp();

x = -x;

}

x = rx;

y = 0;

disp();

p2 = (rx \* rx) + 2.0 \* (ry \* ry \* rx) + (ry \* ry) / 4;

while ((2.0 \* ry \* ry \* x) > (2.0 \* rx \* rx \* y)) {

y++;

if (p2 > 0) p2 = p2 + (rx \* rx) - (2.0 \* rx \* rx \* y);

else {

x--;

p2 = p2 + (2.0 \* ry \* ry \* x) - (2.0 \* rx \* rx \* y) + (rx \* rx);

}

disp(); y = -y;

disp(); y = -y;

}

getch();

closegraph();

}

void disp() {

delay(50);

putpixel(xc + x, yc + y, 10);

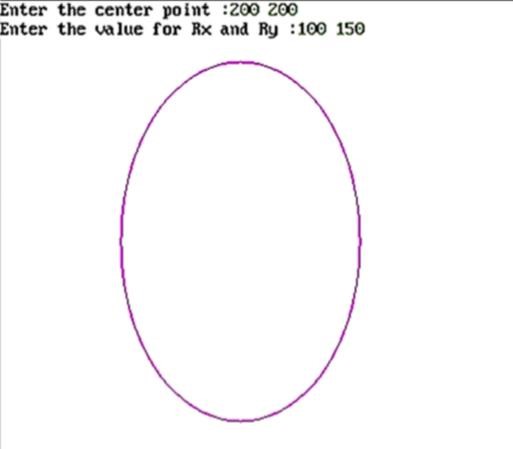
putpixel(xc - x, yc + y, 10);

putpixel(xc + x, yc - y, 10);

putpixel(xc - x, yc - y, 10);

}

OUTPUT:



# 5. WRITE A PROGRAM TO IMPLEMENT 4-CONNECTED FLOOD FILL ALGORITHM

#include<stdio.h>

#include<conio.h>

#include<graphics.h>

#include<dos.h>

void flood(int, int, int, int);

void main() {

int gd, gm = DETECT;

clrscr();

detectgraph( & gd, & gm);

initgraph( & gd, & gm, "../BGI");

rectangle(50, 50, 100, 100);

flood(55, 55, 9, 0);

getch();

}

void flood(int x, int y, int fill\_col, int old\_col) {

if (getpixel(x, y) == old\_col) {

putpixel(x, y, fill\_col);

flood(x + 1, y, fill\_col, old\_col);

flood(x - 1, y, fill\_col, old\_col);

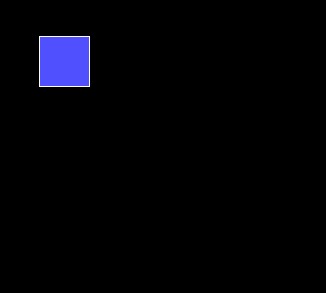
flood(x, y + 1, fill\_col, old\_col);

flood(x, y - 1, fill\_col, old\_col);

}

}

OUTPUT:



# 6. WRITE A PROGRAM TO IMPLEMENT 8-CONNECTED FLOOD FILL ALGORITHM

#include<stdio.h>

#include<conio.h>

#include<graphics.h>

#include<dos.h>

void flood(int, int, int, int);

void main() {

int gd, gm = DETECT;

clrscr();

detectgraph( & gd, & gm);

initgraph( & gd, & gm, "../BGI");

rectangle(50, 50, 100, 100);

flood(55, 55, 9, 0);

getch();

}

void flood(int x, int y, int fill\_col, int old\_col) {

if (getpixel(x, y) == old\_col) {

putpixel(x, y, fill\_col);

flood(x + 1, y, fill\_col, old\_col);

flood(x - 1, y, fill\_col, old\_col);

flood(x, y + 1, fill\_col, old\_col);

flood(x, y - 1, fill\_col, old\_col);

flood(x + 1, y - 1, fill\_col, old\_col);

flood(x + 1, y + 1, fill\_col, old\_col);

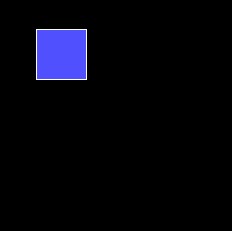
flood(x - 1, y - 1, fill\_col, old\_col);

flood(x - 1, y + 1, fill\_col, old\_col);

}

}

OUTPUT:



# 7. WRITE A PROGRAM TO IMPLEMENT 2D ROTATION

#include<stdio.h>

#include<conio.h>

#include<graphics.h>

#include<math.h>

void main() {

int gd = DETECT, gm;

float x1, y1, x2, y2, x3, y3, x4, y4, a, t;

initgraph( & gd, & gm, "C:\\TurboC3\\BGI");

printf("Enter coordinates of starting point:\n");

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

printf("Enter coordinates of ending point\n");

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

printf("Enter angle for rotation\n");

scanf("%f", & a);

setcolor(5);

line(x1, y1, x2, y2);

outtextxy(x2 + 2, y2 + 2, "Original line");

t = a \* (3.14 / 180);

x3 = (x1 \* cos(t)) - (y1 \* sin(t));

y3 = (x1 \* sin(t)) + (y1 \* cos(t));

x4 = (x2 \* cos(t)) - (y2 \* sin(t));

y4 = (x2 \* sin(t)) + (y2 \* cos(t));

setcolor(7);

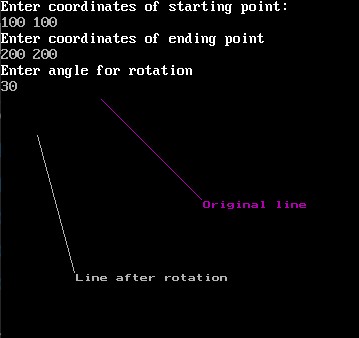
line(x3, y3, x4, y4);

outtextxy(x4 + 2, y4 + 2, "Line after rotation");

getch();

}

OUTPUT:



# 8. WRITE A PROGRAM TO IMPLEMENT 2D TRANSLATION

#include<stdio.h>

#include<conio.h>

#include<graphics.h> #include<math.h> void main(){

int gd = DETECT, gm;

int x1, y1, x2, y2, tx, ty, x3, y3, x4, y4;

initgraph( & gd, & gm, "C:\\TurboC3\\BGI");

printf("Enter the starting point of line segment:");

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

printf("Enter the ending point of line segment:");

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

printf("Enter translation distances tx,ty:\n");

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

setcolor(5);

line(x1, y1, x2, y2);

outtextxy(x2 + 2, y2 + 2, "Original line");

x3 = x1 + tx;

y3 = y1 + ty;

x4 = x2 + tx;

y4 = y2 + ty;

setcolor(7);

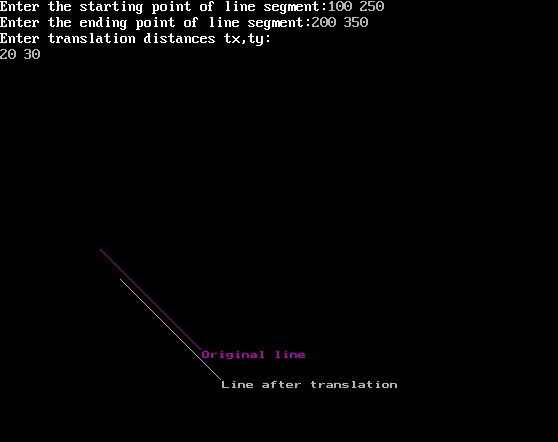
line(x3, y3, x4, y4);

outtextxy(x4 + 2, y4 + 2, "Line after translation");

getch();

}

OUTPUT:



# 9. WRITE A PROGRAM TO IMPLEMENT SCALING

#include<stdio.h>

#include<conio.h>

#include<graphics.h> #include<math.h> void main(){

int gd = DETECT, gm;

float x1, y1, x2, y2, sx, sy, x3, y3, x4, y4;

initgraph( & gd, & gm, "C:\\TurboC3\\BGI");

printf("Enter the starting point coordinates:");

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

printf("Enter the ending point coordinates:");

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

printf("Enter scaling factors sx,sy:\n");

scanf("%f%f", & sx, & sy);

setcolor(5);

line(x1, y1, x2, y2);

outtextxy(x2 + 2, y2 + 2, "Original line");

x3 = x1 \* sx;

y3 = y1 \* sy;

x4 = x2 \* sx;

y4 = y2 \* sy;

setcolor(7);

line(x3, y3, x4, y4);

outtextxy(x4 + 2, y4 + 2, "Line after scaling");

getch();

}

OUTPUT:



# 10. WRITE A PROGRAM TO IMPLEMENT SHEAR-X

#include<stdio.h>

#include<conio.h>

#include<dos.h>

#include<graphics.h>

void main() {

int gd = DETECT, gm;

float shx, shy;

initgraph( & gd, & gm, "C:\\TurboC3\\BGI");

printf("Enter shear factor shx along x-axis :");

scanf("%f", & shx);

line(100, 0, 200, 0);

line(200, 0, 200, 200);

line(200, 200, 100, 200);

line(100, 200, 100, 0);

printf("X-shear");

setcolor(12);

line((100 + (0 \* shx)), 0, (200 + (0 \* shx)), 0);

line((200 + (0 \* shx)), 0, (200 + (200 \* shx)), 200);

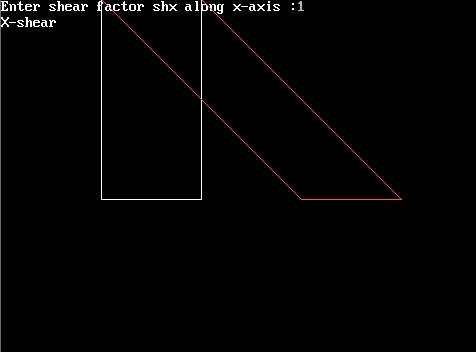
line((200 + (200 \* shx)), 200, (100 + (200 \* shx)), 200);

line((100 + (200 \* shx)), 200, (100 + (0 \* shx)), 0);

getch();

}

OUTPUT:



# 11. WRITE A PROGRAM TO IMPLEMENT SHEAR-Y ALGORITHM

#include<stdio.h>

#include<conio.h>

#include<graphics.h>

void main() {

int gd = DETECT, gm;

int x, y, x1, y1, x2, y2, shear\_f;

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

printf("\n please enter shearing factor y = ");

scanf("%d", & shear\_f);

cleardevice();

line(0, 0, 100, 0);

line(100, 0, 50, 50);

line(50, 50, 0, 0);

setcolor(RED);

y = 0 + 0 \* shear\_f;

y1 = 0 + 100 \* shear\_f;

y2 = 50 + 50 \* shear\_f;

line(0, y, 50, y1);

line(50, y1, 50, y2);

line(50, y2, 0, y);

getch();

closegraph();

}

OUTPUT:



# 12. WRITE A PROGRAM TO IMPLEMENT LINE CLIPPING ALGORITHM

#include<conio.h>

#include<stdio.h>

#include<graphics.h>

#include<math.h>

void main() {

int a[4], b[4];

float m, xnew, ynew;

float xl = 100, yl = 100, xh = 300, yh = 300, xa = 10, ya = 200, xb = 250, yb = 150;

int gd = DETECT, gm;

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

setcolor(5);

line(xa, ya, xb, yb);

setcolor(12);

rectangle(xl, yl, xh, yh);

m = (yb - ya) / (xb - xa);

if (xa < xl) a[3] = 1;

else a[3] = 0;

if (xa > xh) a[2] = 1;

else a[2] = 0;

if (ya < yl) a[1] = 1;

else a[1] = 0;

if (ya > yh) a[0] = 1;

else a[0] = 0;

if (xb < xl) b[3] = 1;

else b[3] = 0;

if (xb > xh) b[2] = 1;

else b[2] = 0;

if (yb < yl) b[1] = 1;

else b[1] = 0;

if (yb > yh) b[0] = 1;

else b[0] = 0;

printf("press a key to continue");

getch();

if (a[0] == 0 && a[1] == 0 && a[2] == 0 && a[3] == 0 && b[0] == 0 && b[1] == 0 && b[2] == 0 && b[3] == 0) {

printf("no clipping");

line(xa, ya, xb, yb);

} else if (a[0] && b[0] || a[1] && b[1] || a[2] && b[2] || a[3] && b[3]) {

clrscr();

printf("line discarded");

rectangle(xl, yl, xh, yh);

} else {

if (a[3] == 1 && b[3] == 0) {

ynew = (m \* (xl - xa)) + ya;

setcolor(12);

rectangle(xl, yl, xh, yh);

setcolor(0);

line(xa, ya, xb, yb);

setcolor(15);

line(xl, ynew, xb, yb);

} else if (a[2] == 1 && b[2] == 0) {

ynew = (m \* (xh - xa)) + ya;

setcolor(12);

rectangle(xl, yl, xh, yh);

setcolor(0);

line(xa, ya, xb, yb);

setcolor(15);

line(xl, ynew, xb, yb);

} else if (a[1] == 1 && b[1] == 0) {

xnew = xa + (yl - ya) / m;

setcolor(0);

line(xa, ya, xb, yb);

setcolor(15);

line(xnew, yh, xb, yb);

} else if (a[0] == 1 && b[0] == 0) {

xnew = xa + (yh - ya) / m;

setcolor(0);

line(xa, ya, xb, yb);

setcolor(15);

line(xnew, yh, xb, yb);

}

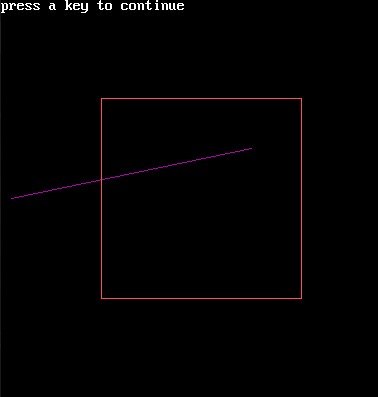
}

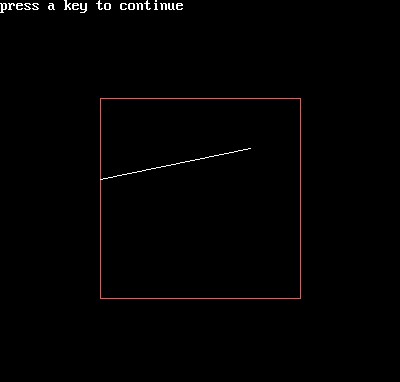
getch();

closegraph();

}

OUTPUT:





# 13. WRITE A PROGRAM TO IMPLEMENT 3D ROTATION

#include<stdio.h>

#include<conio.h>

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

int x1, x2, y1, y2, mx, my, depth;

void draw() {

bar3d(x1, y1, x2, y2, depth, 1);

}

void rotate() {

float t;

int a1, b1, a2, b2, dep;

printf("Enter the angle to rotate=");

scanf("%f", & t);

t = t \* (3.14 / 180);

a1 = mx + (x1 - mx) \* cos(t) - (y1 - my) \* sin(t);

a2 = mx + (x2 - mx) \* cos(t) - (y2 - my) \* sin(t);

b1 = my + (x1 - mx) \* sin(t) - (y1 - my) \* cos(t);

b2 = my + (x2 - mx) \* sin(t) - (y2 - my) \* cos(t);

if (a2 > a1) dep = (a2 - a1) / 4;

else

dep = (a1 - a2) / 4;

bar3d(a1, b1, a2, b2, dep, 1);

setcolor(5);

}

void main() {

int gd = DETECT, gm, c;

initgraph( & gd, & gm, "C:\\TurboC3\\BGI");

printf("\n3D Transformation Rotating\n\n");

printf("\nEnter 1st top value(x1,y1):");

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

printf("Enter right bottom value(x2,y2):");

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

depth = (x2 - x1) / 4;

mx = (x1 + x2) / 2;

my = (y1 + y2) / 2;

draw();

getch();

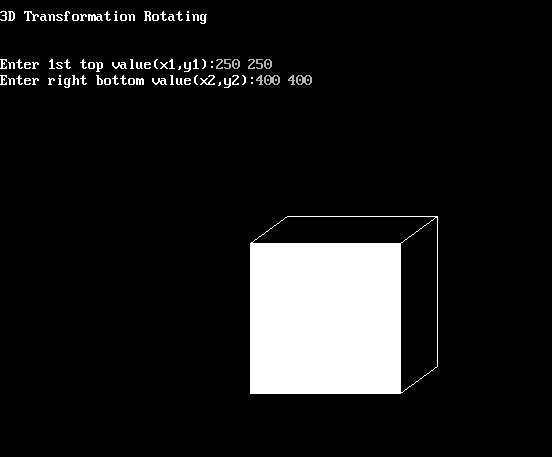
cleardevice();

rotate();

getch();

}

OUTPUT:



# 14. WRITE A PROGRAM TO IMPLEMENT 3D SCALING

#include<stdio.h>

#include<conio.h>

#include<math.h>

#include<process.h> #include<graphics.h> int x1,x2,y1,y2,mx,my,depth;

void draw() {

bar3d(x1, y1, x2, y2, depth, 1);

}

void scale() {

int x, y, a1, a2, b1, b2, dep;

printf("\n\n Enter scaling Factors:");

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

a1 = mx + (x1 - mx) \* x;

a2 = mx + (x2 - mx) \* x;

b1 = my + (y1 - my) \* y;

b2 = my + (y2 - my) \* y;

dep = (a2 - a1) / 4;

bar3d(a1, b1, a2, b2, dep, 1);

setcolor(5);

draw();

}

void main() {

int gd = DETECT, gm, c;

initgraph( & gd, & gm, "C:\\TurboC3\\BGI");

printf("\n\t\t3D Scaling\n\n");

printf("\nEnter 1st top value(x1,y1):");

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

printf("Enter right bottom value(x2,y2):");

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

depth = (x2 - x1) / 4;

mx = (x1 + x2) / 2;

my = (y1 + y2) / 2;

draw();

getch();

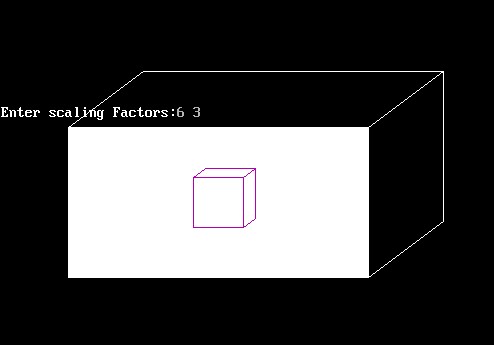
cleardevice();

scale();

getch();

}

OUTPUT:



# 15. WRITE A PROGRAM TO IMPLEMENT 3-D TRANSLATION

#include<stdio.h>

#include<conio.h>

#include<math.h>

#include<process.h>

#include<graphics.h>

int x1, x2, y1, y2, mx, my, depth;

void draw();

void trans();

void main() {

int gd = DETECT, gm, c;

initgraph( & gd, & gm, "C:\\TurboC3\\BGI");

printf("\n\t\t3D Translation\n\n");

printf("\nEnter 1st top value(x1,y1):");

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

printf("Enter right bottom value(x2,y2):");

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

depth = (x2 - x1) / 4;

mx = (x1 + x2) / 2;

my = (y1 + y2) / 2;

draw();

getch();

cleardevice();

trans();

getch();

}

void draw() {

bar3d(x1, y1, x2, y2, depth, 1);

}

void trans() {

int a1, a2, b1, b2, dep, x, y;

printf("\n Enter the Translation Distances:");

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

a1 = x1 + x;

a2 = x2 + x;

b1 = y1 + y;

b2 = y2 + y;

dep = (a2 - a1) / 4;

bar3d(a1, b1, a2, b2, dep, 1);

setcolor(5);

draw();

}

OUTPUT:

