**Vedas College**



**Lab Report of**

**COMPUTER GRAPHICS**

**Submitted by:**  **Submitted to:**

Rajpati Bhandari Suresh Thapa

**Date:**

**INDEX**

|  |  |  |  |
| --- | --- | --- | --- |
| **Lab No.** | **Content** | **Date** | **Signature** |
| **1.** | **Draw a line**  1. Using the DDA algorithm 2. Using Bresenham algorithm |  |  |
| **2.** | **Draw a circle**  1. Using the Direct Method 2. Using the Trigonometric method 3. Using the Mid-point circle method |  |  |
| **3.** | **Draw an ellipse**Using the Direct method  1. Using the Trigonometric method 2. Using the Mid-Point ellipse method |  |  |
| **4.** | **2D-geometrical transformations**   1. Translation 2. Rotation 3. Rotation about an arbitrary point 4. Scaling 5. Reflection 6. X-shear 7. Y-shear |  |  |

# **Lab no. 1 – Draw a line**

## **Using the Digital Differential Analyzer (DDA) algorithm**

#include <stdio.h>

#include <graphics.h>

#include <math.h>

void drawLineDDA(int x1, int y1, int x2, int y2) {

int gd = DETECT, gm;

initgraph(&gd, &gm, NULL);

int dx = x2 - x1;

int dy = y2 - y1;

int steps = abs(dx) > abs(dy) ? abs(dx) : abs(dy);

float xIncrement = dx / (float)steps;

float yIncrement = dy / (float)steps;

float x = x1;

float y = y1;

for (int i = 0; i <= steps; i++) {

putpixel(round(x), round(y), WHITE);

x += xIncrement;

y += yIncrement;

}

delay(5000);

closegraph();

}

int main() {

int x1, y1, x2, y2;

printf("\n\*\*\*\*\*\* DDA Line Drawing Algorithm \*\*\*\*\*\*\*\*\*\*\*");

printf("Enter the coordinates of the first point (x1, y1): ");

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

printf("Enter the coordinates of the second point (x2, y2): ");

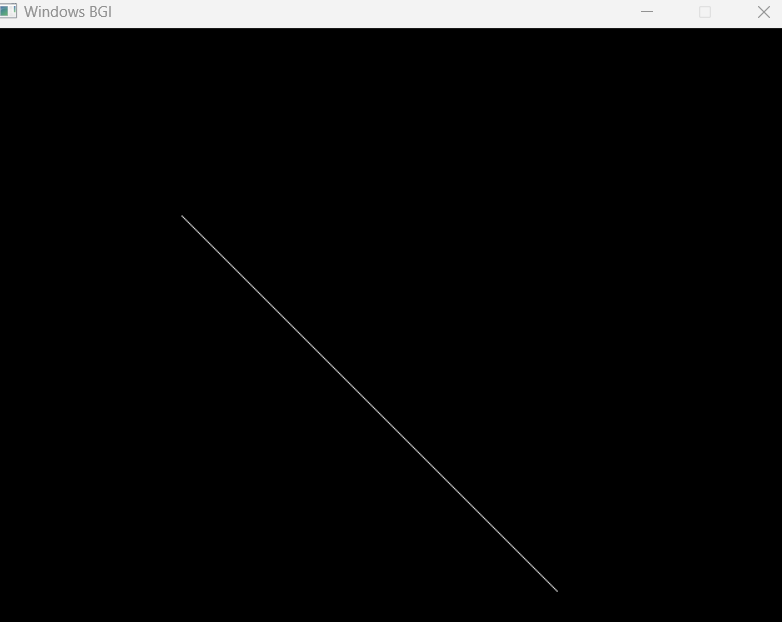
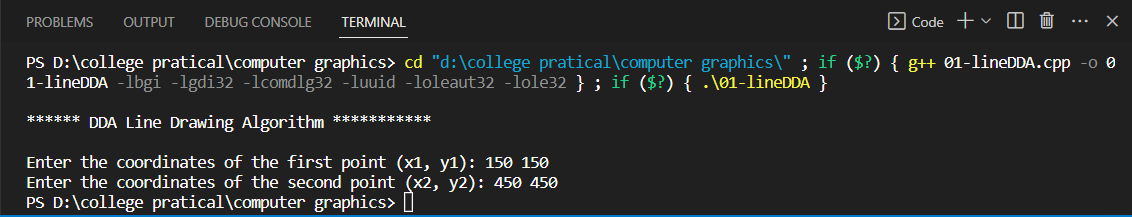
scanf("%d %d", &x2, &y2);

drawLineDDA(x1, y1, x2, y2);

return 0;

}

**Output:**



## 

## **Using Bresenham algorithm**

#include <stdio.h>

#include <graphics.h>

#include <math.h>

void drawLineBresenham(int x1, int y1, int x2, int y2) {

int gd = DETECT, gm;

initgraph(&gd, &gm, NULL);

int dx = abs(x2 - x1);

int dy = abs(y2 - y1);

int x = x1;

int y = y1;

int xIncrement = (x1 < x2) ? 1 : -1;

int yIncrement = (y1 < y2) ? 1 : -1;

int interchange = 0;

if (dy > dx) {

int temp = dx;

dx = dy;

dy = temp;

interchange = 1;

}

int error = 2 \* dy - dx;

for (int i = 0; i <= dx; i++) {

putpixel(x, y, WHITE);

while (error >= 0) {

if (interchange)

x += xIncrement;

else

y += yIncrement;

error -= 2 \* dx;

}

if (interchange)

y += yIncrement;

else

x += xIncrement;

error += 2 \* dy;

}

delay(5000);

closegraph();

}

int main() {

int x1, y1, x2, y2;

printf("\*\*\*\*\*\*\* BRESENHAM'S LINE DRAWING ALGORITHM \*\*\*\*\*\n\n");

printf("Enter the coordinates of the first point (x1, y1): ");

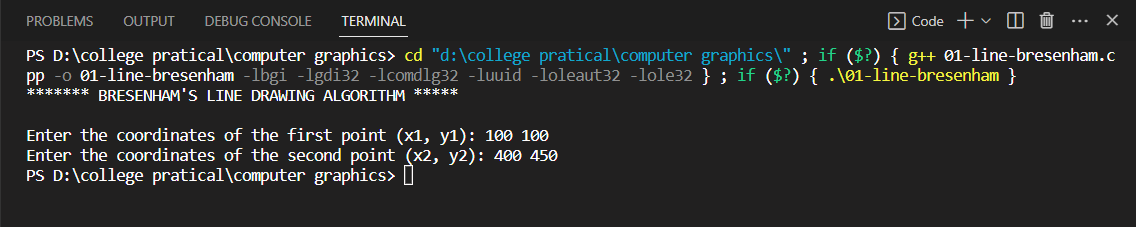
scanf("%d %d", &x1, &y1);

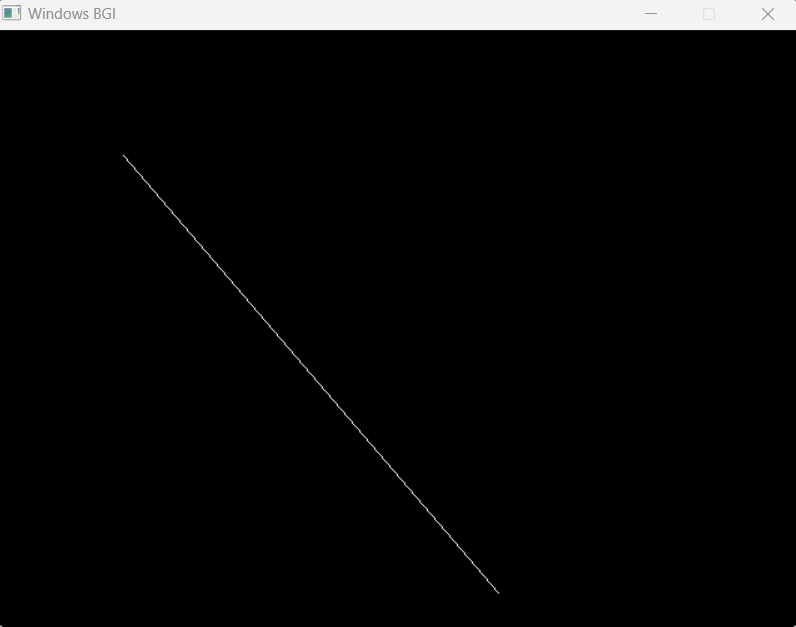
printf("Enter the coordinates of the second point (x2, y2): ");

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

drawLineBresenham(x1, y1, x2, y2);

return 0;

**Output:**



# **Lab no. 2 - Draw a circle**

## **Using the Direct Method**

#include <stdio.h>

#include <graphics.h>

void drawCircle(int centerX, int centerY, int radius) {

int gd = DETECT, gm;

initgraph(&gd, &gm, NULL);

int x = 0;

int y = radius;

int decision = 3 - (2 \* radius);

while (x <= y) {

putpixel(centerX + x, centerY + y, WHITE);

putpixel(centerX - x, centerY + y, WHITE);

putpixel(centerX + x, centerY - y, WHITE);

putpixel(centerX - x, centerY - y, WHITE);

putpixel(centerX + y, centerY + x, WHITE);

putpixel(centerX - y, centerY + x, WHITE);

putpixel(centerX + y, centerY - x, WHITE);

putpixel(centerX - y, centerY - x, WHITE);

if (decision <= 0) {

decision += (4 \* x) + 6;

} else {

decision += (4 \* (x - y)) + 10;

y--;

}

x++;

}

delay(5000);

closegraph();

}

int main() {

int centerX, centerY, radius;

printf("\n\*\*\*\*\*\*\* DIRECT CIRCLE DRAWING ALGORITHM \*\*\*\*\*\n\n");

printf("Enter the center coordinates of the circle (centerX, centerY): ");

scanf("%d %d", &centerX, &centerY);

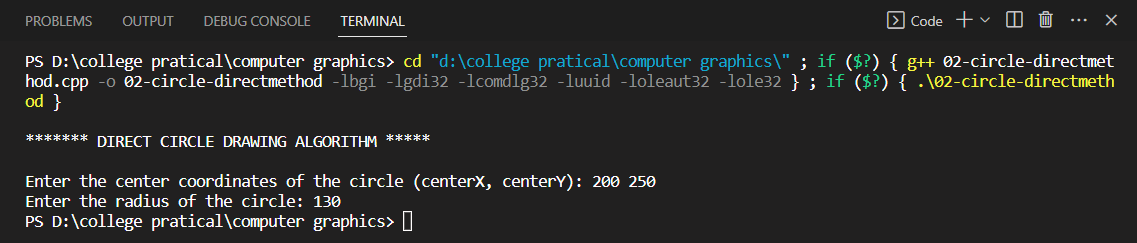
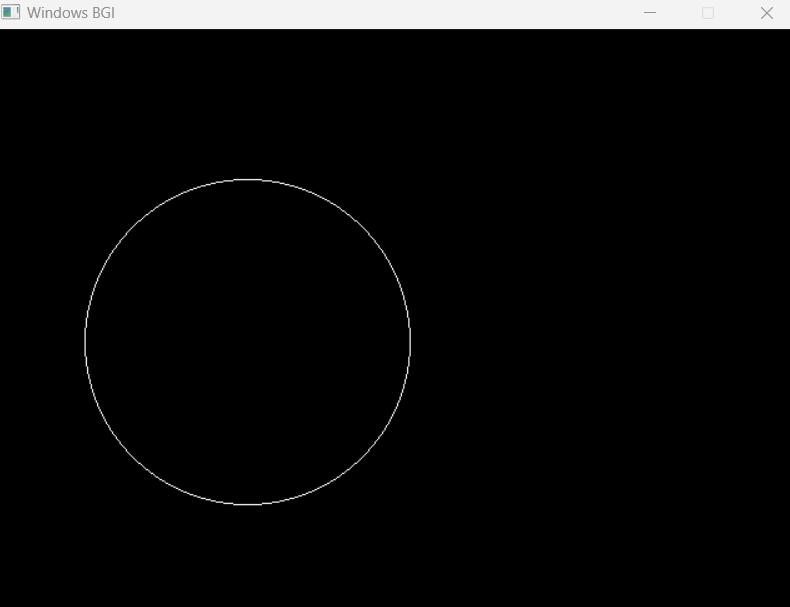
printf("Enter the radius of the circle: ");

scanf("%d", &radius)

drawCircle(centerX, centerY, radius);

return 0;

}  
  
**Output:**

****

## **Using the Trigonometric method**

#include <stdio.h>

#include <math.h>

#include <graphics.h>

void drawCircle(int centerX, int centerY, int radius) {

int gd = DETECT, gm;

initgraph(&gd, &gm, NULL);

float angle = 0;

float angleIncrement = 0.01;

while (angle <= 2 \* M\_PI) {

int x = centerX + round(radius \* cos(angle));

int y = centerY + round(radius \* sin(angle));

putpixel(x, y, WHITE);

angle += angleIncrement;

}

delay(5000);

closegraph();

}

int main() {

int centerX, centerY, radius;

    printf("\n\*\*\*\*\*\*\* CIRCLE DRAWING TRIGONOMETRIC METHOD ALGORITHM \*\*\*\*\*\n\n");

printf("Enter the center coordinates of the circle (centerX, centerY): ");

scanf("%d %d", &centerX, &centerY);

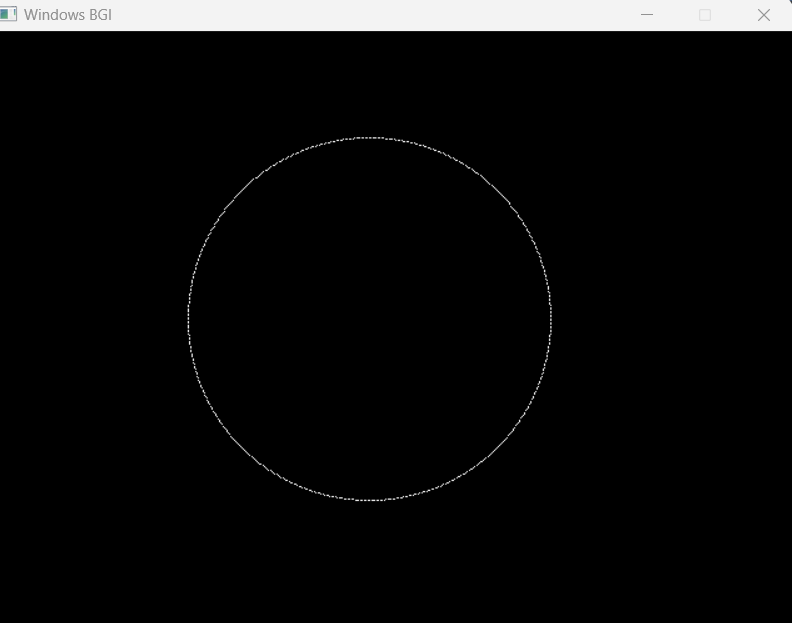
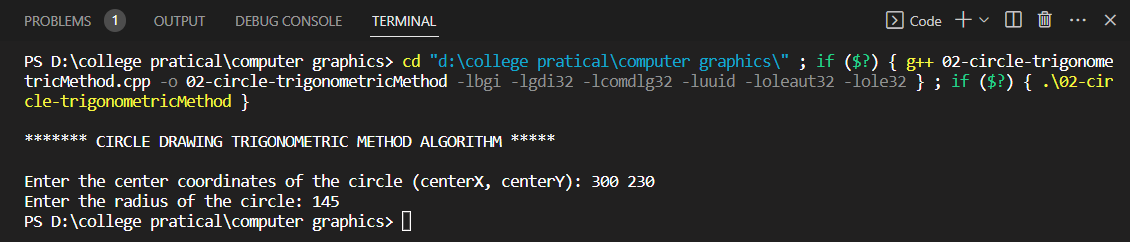
printf("Enter the radius of the circle: ");

scanf("%d", &radius);

drawCircle(centerX, centerY, radius);

return 0;

}  
  
  
**Output:**



## **Using the Mid-point circle method**

#include <stdio.h>

#include <graphics.h>

void drawCircle(int centerX, int centerY, int radius) {

int gd = DETECT, gm;

initgraph(&gd, &gm, NULL);

int x = 0;

int y = radius;

int decision = 1 - radius;

while (x <= y) {

putpixel(centerX + x, centerY + y, WHITE);

putpixel(centerX - x, centerY + y, WHITE);

putpixel(centerX + x, centerY - y, WHITE);

putpixel(centerX - x, centerY - y, WHITE);

putpixel(centerX + y, centerY + x, WHITE);

putpixel(centerX - y, centerY + x, WHITE);

putpixel(centerX + y, centerY - x, WHITE);

putpixel(centerX - y, centerY - x, WHITE);

x++;

if (decision < 0) {

decision += 2 \* x + 1;

} else {

y--;

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

}

}

delay(5000);

closegraph();

}

int main()

{

int centerX, centerY, radius;

printf("\*\*\*\*\*\*\*\*\* MID POINT Circle drawing algorithm \*\*\*\*\*\*\*\*\n\n");

printf("Enter the center coordinates of the circle (centerX, centerY): ");

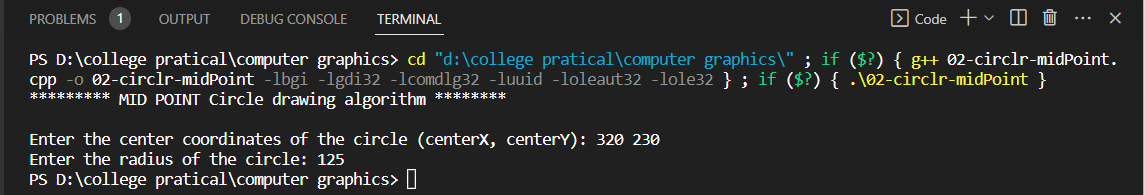
scanf("%d %d", &centerX, &centerY);

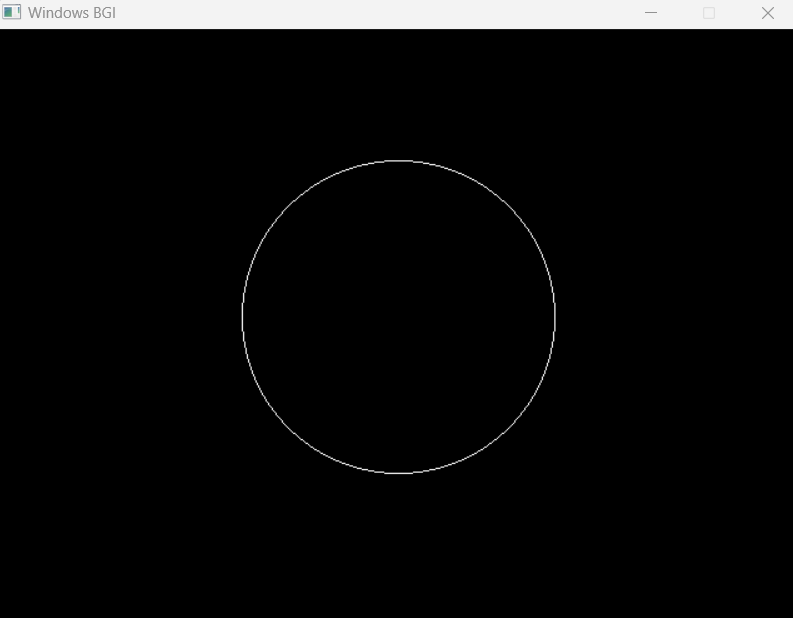
printf("Enter the radius of the circle: ");

scanf("%d", &radius);

drawCircle(centerX, centerY, radius);

return 0;

}  
  
  
**Output:**



# **Lab no. 3 - Draw an ellipse**

## **Using Direct method**

#include <stdio.h>

#include <math.h>

#include <graphics.h>

void drawEllipse(int centerX, int centerY, int semiMajorAxis, int semiMinorAxis) {

int gd = DETECT, gm;

initgraph(&gd, &gm, NULL);

int aSq = semiMajorAxis \* semiMajorAxis;

int bSq = semiMinorAxis \* semiMinorAxis;

int x = 0;

int y = semiMinorAxis;

int decision = bSq + (aSq \* (0.25 - semiMinorAxis));

while ((2 \* bSq \* x) <= (2 \* aSq \* y)) {

putpixel(centerX + x, centerY - y, WHITE);

putpixel(centerX - x, centerY - y, WHITE);

putpixel(centerX + x, centerY + y, WHITE);

putpixel(centerX - x, centerY + y, WHITE);

x++;

if (decision < 0) {

decision += (2 \* bSq \* x) + bSq;

} else {

y--;

decision += (2 \* bSq \* x) - (2 \* aSq \* y) + bSq;

}

}

decision = bSq \* (x + 0.5) \* (x + 0.5) + (aSq \* (y - 1) \* (y - 1)) - (aSq \* bSq);

while (y >= 0) {

putpixel(centerX + x, centerY - y, WHITE);

putpixel(centerX - x, centerY - y, WHITE);

putpixel(centerX + x, centerY + y, WHITE);

putpixel(centerX - x, centerY + y, WHITE);

y--;

if (decision > 0) {

decision += aSq - (2 \* aSq \* y);

} else {

x++;

decision += (2 \* bSq \* x) - (2 \* aSq \* y) + aSq;

}

}

delay(5000);

closegraph();

}

int main() {

int centerX, centerY, semiMajorAxis, semiMinorAxis;

printf("Enter the center coordinates of the ellipse (centerX, centerY): ");

scanf("%d %d", &centerX, &centerY);

printf("Enter the semi-major axis length: ");

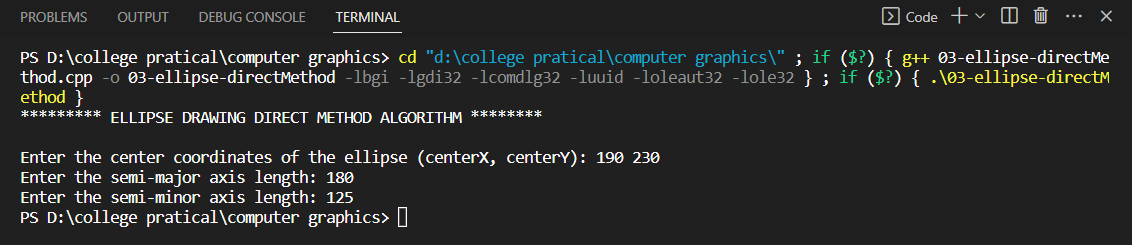
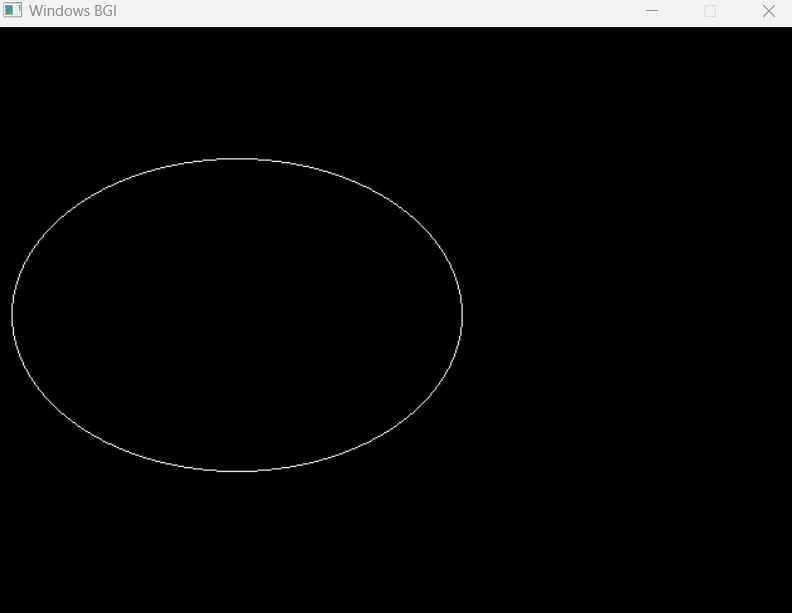
scanf("%d", &semiMajorAxis);

printf("Enter the semi-minor axis length: ");

scanf("%d", &semiMinorAxis);

drawEllipse(centerX, centerY, semiMajorAxis, semiMinorAxis);

return 0;

}  
  
  
**Output:**

## **Using the Trigonometric method**

#include <stdio.h>

#include <math.h>

#include <graphics.h>

void drawEllipse(int centerX, int centerY, int semiMajorAxis, int semiMinorAxis) {

int gd = DETECT, gm;

initgraph(&gd, &gm, NULL);

float angle = 0;

float angleIncrement = 0.01;

while (angle <= 2 \* M\_PI) {

int x = centerX + round(semiMajorAxis \* cos(angle));

int y = centerY + round(semiMinorAxis \* sin(angle));

putpixel(x, y, WHITE);

angle += angleIncrement;

}

delay(5000);

closegraph();

}

int main() {

int centerX, centerY, semiMajorAxis, semiMinorAxis;

  printf("\*\*\*\*\*\*\*\*\* ELLIPSE DRAWING TRIGONOMETRIC METHOD ALGORITHM \*\*\*\*\*\*\*\*\n\n");

printf("Enter the center coordinates of the ellipse (centerX, centerY): ");

scanf("%d %d", &centerX, &centerY);

printf("Enter the semi-major axis length: ");

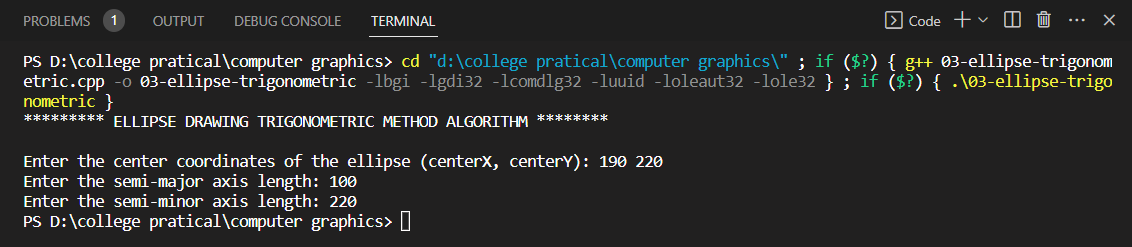
scanf("%d", &semiMajorAxis);

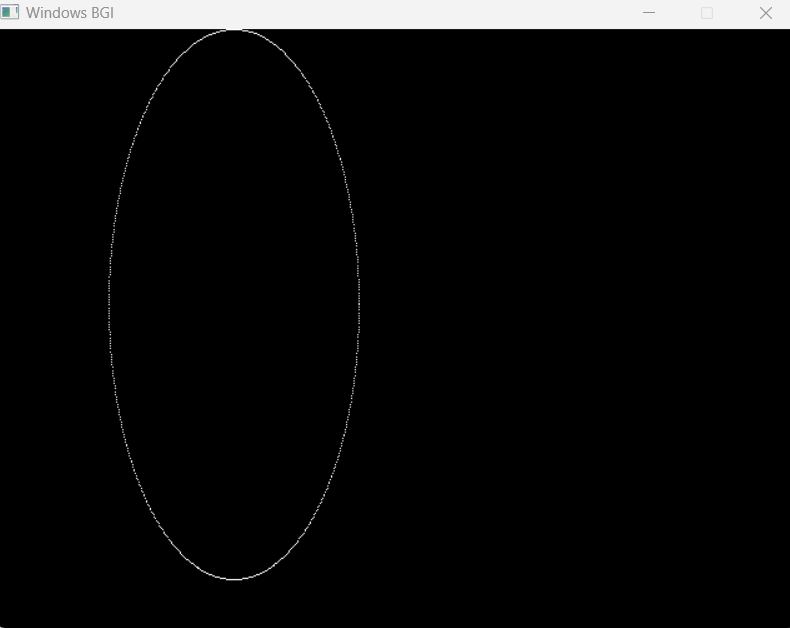
printf("Enter the semi-minor axis length: ");

scanf("%d", &semiMinorAxis);

drawEllipse(centerX, centerY, semiMajorAxis, semiMinorAxis);

return 0;

}  
  
  
**Output:**



## **Using the Mid-Point ellipse method**

#include <stdio.h>

#include<cmath>

#include <graphics.h>

void drawEllipse(int centerX, int centerY, int semiMajorAxis, int semiMinorAxis) {

int gd = DETECT, gm;

initgraph(&gd, &gm, NULL);

int x = 0;

int y = semiMinorAxis;

int aSq = semiMajorAxis \* semiMajorAxis;

int bSq = semiMinorAxis \* semiMinorAxis;

int twoASq = 2 \* aSq;

int twoBSq = 2 \* bSq;

int xEnd = round(aSq / sqrt(aSq + bSq));

int px = 0;

int py = twoASq \* y;

// Region 1

int p = round(bSq - aSq \* semiMinorAxis + 0.25 \* aSq);

while (px <= py) {

putpixel(centerX + x, centerY + y, WHITE);

putpixel(centerX - x, centerY - y, WHITE);

putpixel(centerX + x, centerY - y, WHITE);

putpixel(centerX - x, centerY + y, WHITE);

x++;

px += twoBSq;

if (p < 0) {

p += bSq + px;

} else {

y--;

py -= twoASq;

p += bSq + px - py;

}

}

// Region 2

p = round(bSq \* (x + 0.5) \* (x + 0.5) + aSq \* (y - 1) \* (y - 1) - aSq \* bSq);

while (y >= 0) {

putpixel(centerX + x, centerY + y, WHITE);

putpixel(centerX - x, centerY - y, WHITE);

putpixel(centerX + x, centerY - y, WHITE);

putpixel(centerX - x, centerY + y, WHITE);

y--;

py -= twoASq;

if (p > 0) {

p += aSq - py;

} else {

x++;

px += twoBSq;

p += aSq - py + px;

}

}

delay(5000);

closegraph();

}

int main() {

int centerX, centerY, semiMajorAxis, semiMinorAxis;

    printf("\n\*\*\*\*\*\*\*\*\* ELLIPSE DRAWING MID POINT ALGORITHM \*\*\*\*\*\*\*\*\n\n");

printf("Enter the center coordinates of the ellipse (centerX, centerY): ");

scanf("%d %d", &centerX, &centerY);

printf("Enter the semi-major axis length: ");

scanf("%d", &semiMajorAxis);

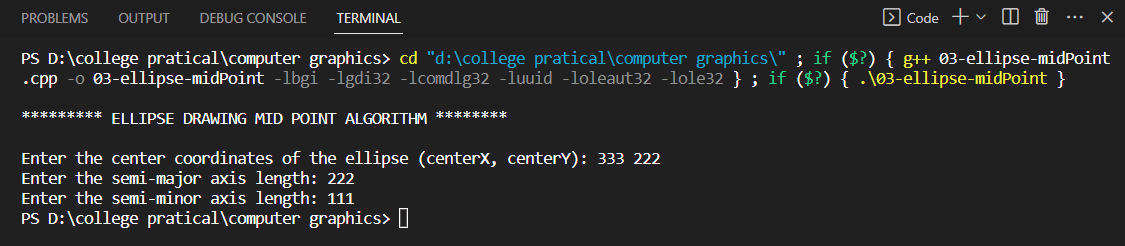
printf("Enter the semi-minor axis length: ");

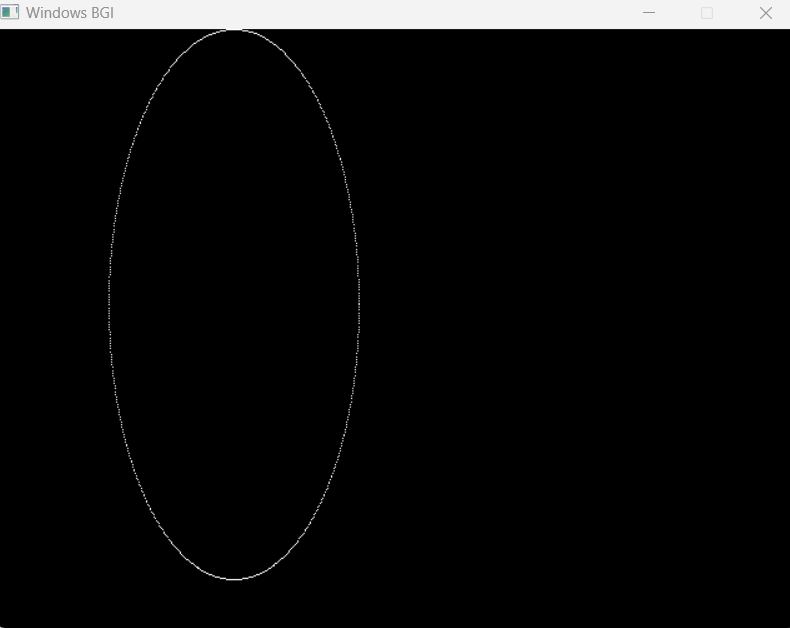
scanf("%d", &semiMinorAxis);

drawEllipse(centerX, centerY, semiMajorAxis, semiMinorAxis);

return 0;

}  
  
  
**Output:**





**2D-geometrical transformations**

1. **Translation**
2. **Rotation**
3. **Rotation about an arbitrary point**
4. **Scaling**

#include <stdio.h>

#include <conio.h>

#include <graphics.h>

#include <math.h>

void translation(int, int, int, int, int, int);

void rotation(int, int, int, int, int);

void rotationArbitrary(int, int, int, int, int, int, int);

void scaling(int, int, int, int, int, int);

int main()

{

int gd = DETECT, gm;

initgraph(&gd, &gm, NULL);

int ch;

printf("\n\n 1. Translation \n 2. Rotation \n 3. Rotation about arbitrary point \n 4. Scaling \n");

printf("Enter your Choice (1,2,3,4): ");

scanf("%d", &ch);

if (ch == 1)

translation(100, 200, 200, 400, 50, 10);

else if (ch == 2)

rotation(50, 200, 50, 300, -45);

else if (ch == 3)

rotationArbitrary(50, 100, 50, 300, 10, 20, -45);

else if (ch == 4)

scaling(50, 100, 65, 200, 5, 2);

else

printf("\n\nInvalid Entry");

getch();

return 0;

}

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

{

outtextxy(x1 - 20, y1 - 20, "Origninal Line");

line(x1, y1, x2, y2);

int x3 = x1 + tx, y3 = y1 + ty;

int x4 = x2 + tx, y4 = y2 + ty;

outtextxy(x4 + 2, y4 + 2, "translated Line");

line(x3, y3, x4, y4);

}

void rotation(int x1, int y1, int x2, int y2, int ang)

{

outtextxy(x1 - 20, y1 - 20, "Origninal Line");

line(x1, y1, x2, y2);

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

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

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

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

outtextxy(x4 + 2, y4 + 2, "rotated Line");

line(x3, y3, x4, y4);

}

void rotationArbitrary(int x1, int y1, int x2, int y2, int xr, int yr, int ang)

{

outtextxy(x1 - 20, y1 - 20, "Origninal Line");

line(x1, y1, x2, y2);

int x3 = xr + (x1 - xr) \* cos(ang) - (y1 - yr) \* sin(ang);

int y3 = yr + (x1 - xr) \* sin(ang) + (y1 - yr) \* cos(ang);

int x4 = xr + (x2 - xr) \* cos(ang) - (y2 - yr) \* sin(ang);

int y4 = yr + (x2 - xr) \* sin(ang) + (y2 - yr) \* cos(ang);

outtextxy(x4 + 2, y4 + 2, "rotated Line about arbitrary point");

line(x3, y3, x4, y4);

}

void scaling(int x1, int y1, int x2, int y2, int sx, int sy)

{

outtextxy(x1 - 20, y1 - 20, "Origninal Line");

line(x1, y1, x2, y2);

int x3 = x1 \* sx, y3 = y1 \* sy;

int x4 = x2 \* sx, y4 = y2 \* sy;

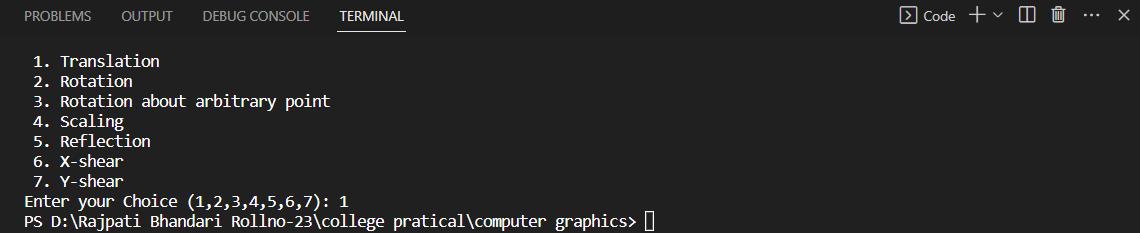
outtextxy(x4 + 2, y4 + 2, "translated Line");

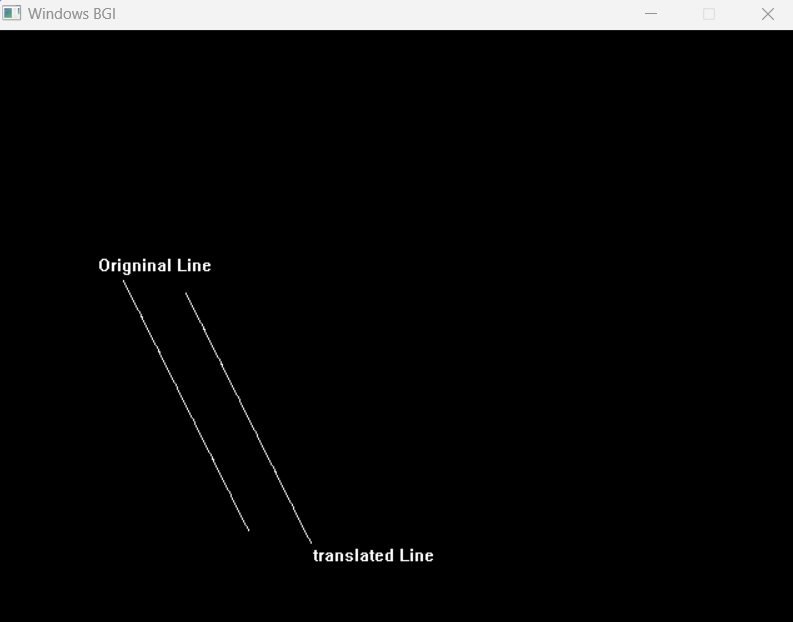
line(x3, y3, x4, y4);

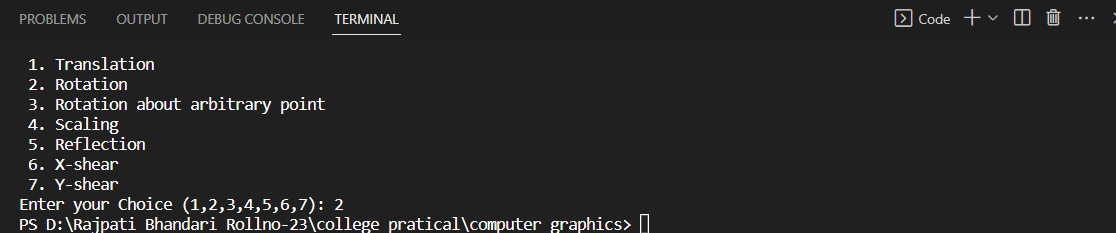
}

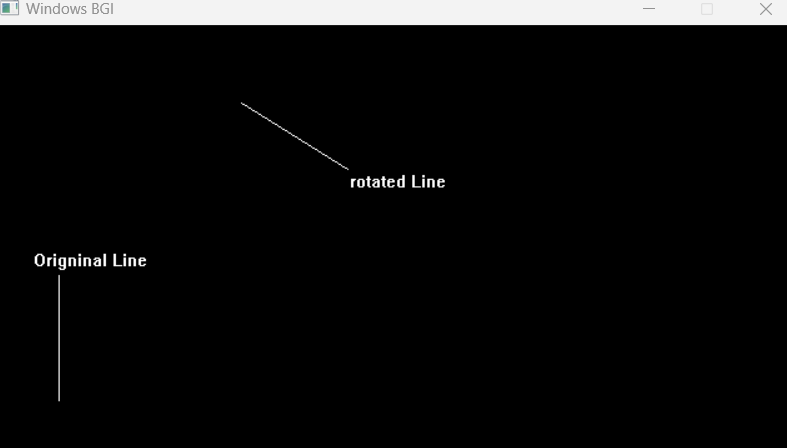
**Outputs:**

Translation

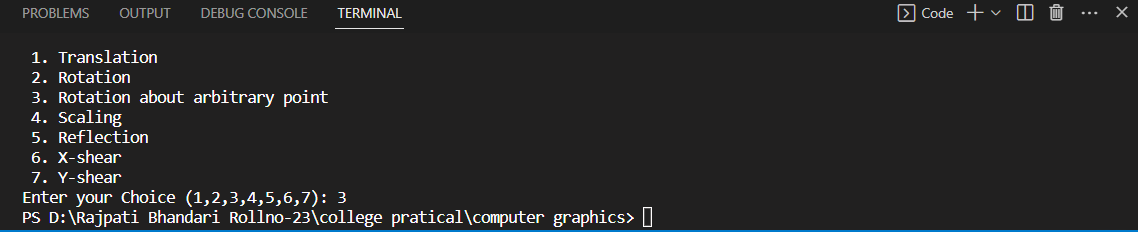


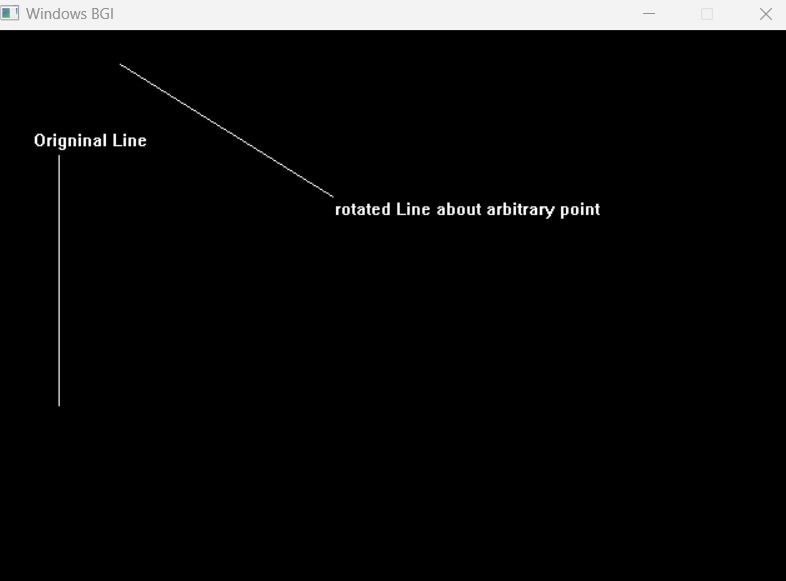


Rotation

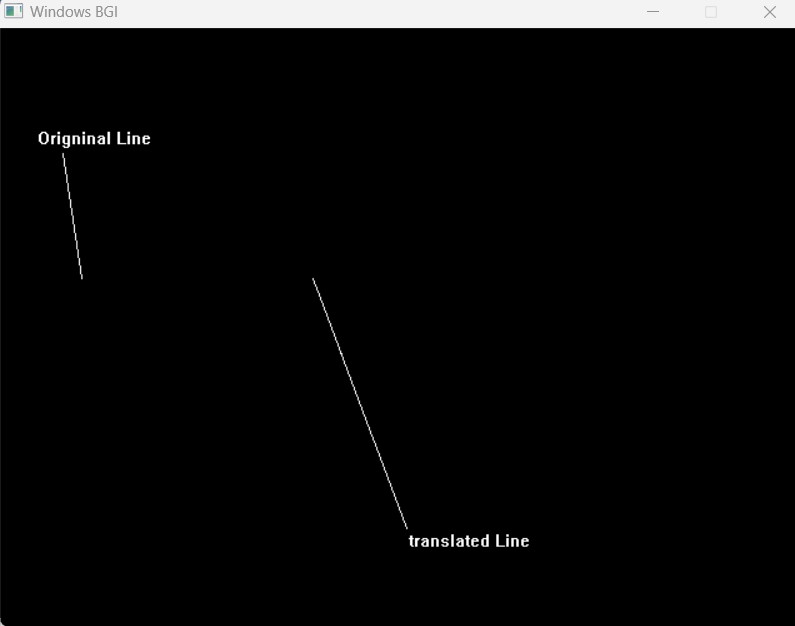
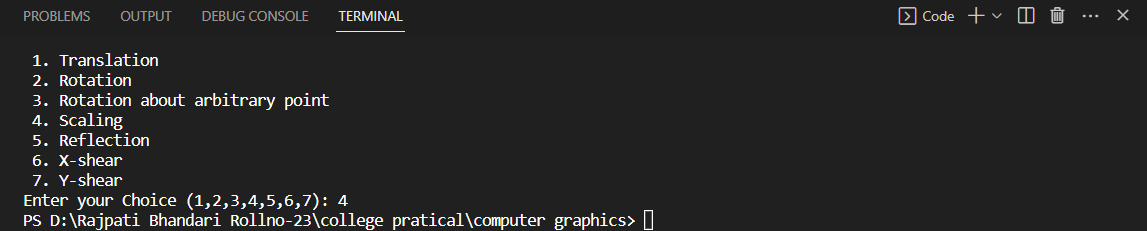


Rotation about an arbitrary point





Scaling



1. **Reflection**

#include <stdio.h>

#include <graphics.h>

#include <math.h>

#define PI acos(-1)

int main()

{

int gd = 0, gm, x1, y1, x2, y2, x3, y3;

double s, c, angle;

initgraph(&gd, &gm, NULL);

printf("Enter coordinates of line: ");

scanf("%d%d%d%d%d%d", &x1, &y1, &x2, &y2, &x3, &y3);

cleardevice();

line(x1, y1, x2, y2);

line(x2, y2, x3, y3);

line(x3, y3, x1, y1);

getch();

printf("Enter rotation angle: ");

scanf("%lf", &angle);

c = cos(angle \* PI / 180);

s = sin(angle \* PI / 180);

x1 = floor(x1 \* c + y1 \* s);

y1 = floor(-x1 \* s + y1 \* c);

x2 = floor(x2 \* c + y2 \* s);

y2 = floor(-x2 \* s + y2 \* c);

x3 = floor(x3 \* c + y3 \* s);

y3 = floor(-x3 \* s + y3 \* c);

printf("%d %d %d %d", x1, y1, x2, y2);

cleardevice();

line(x1, y1, x2, y2);

line(x2, y2, x3, y3);

line(x3, y3, x1, y1);

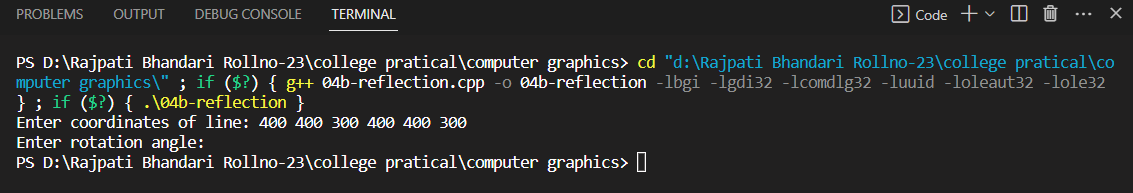
getch();

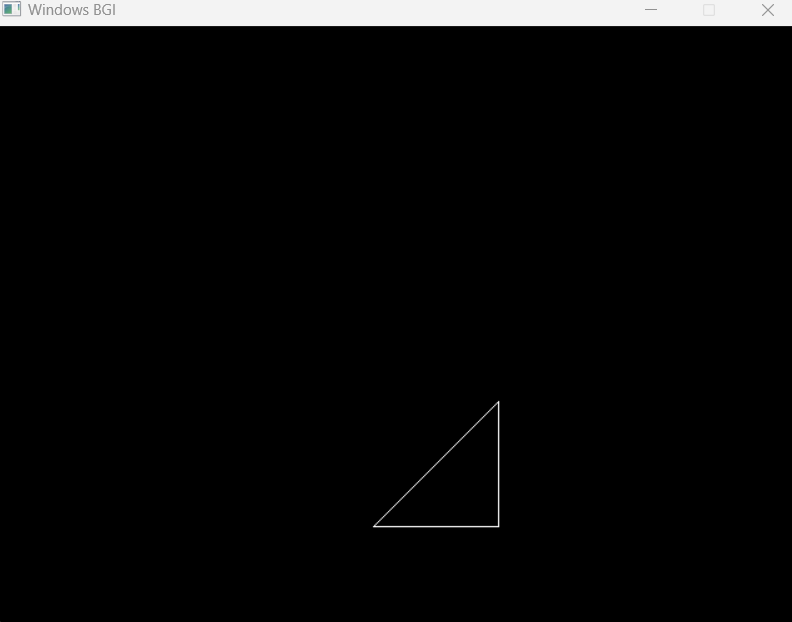
closegraph();

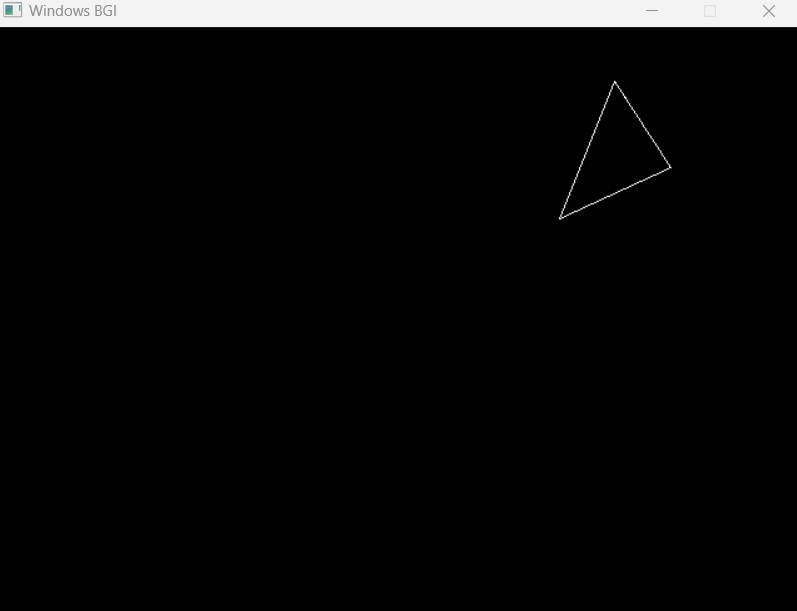
return 0;

}

**Output:**

****

****

****

1. **X-shear**

#include <stdio.h>

#include <graphics.h>

#include <conio.h>

int main()

{

int gd = DETECT, gm;

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

initgraph(&gd, &gm, NULL);

printf("\nplease enter first coordinate = ");

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

printf("please enter second coordinate = ");

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

printf("please enter third coordinate = ");

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

printf("please enter shearing factor x = ");

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

cleardevice();

line(x, y, x1, y1);

line(x1, y1, x2, y2);

line(x2, y2, x, y);

setcolor(WHITE);

x = x + y \* shear\_f;

x1 = x1 + y1 \* shear\_f;

x2 = x2 + y2 \* shear\_f;

line(x, y, x1, y1);

line(x1, y1, x2, y2);

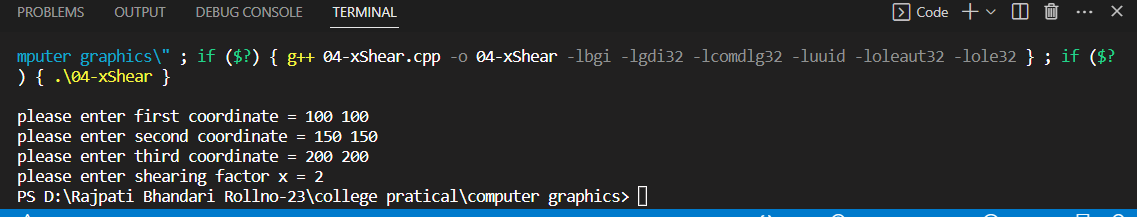
line(x2, y2, x, y);

getch();

closegraph();

return 0;

}

**Output:**

****

1. **Y-shear**

#include <stdio.h>

#include <graphics.h>

#include <conio.h>

int main()

{

int gd = DETECT, gm;

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

initgraph(&gd, &gm, NULL);

printf("\nplease enter first coordinate = ");

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

printf("please enter second coordinate = ");

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

printf("please enter third coordinate = ");

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

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

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

cleardevice();

line(x, y, x1, y1);

line(x1, y1, x2, y2);

line(x2, y2, x, y);

setcolor(WHITE);

y = y + x \* shear\_f;

y1 = y1 + x1 \* shear\_f;

y2 = y2 + x2 \* shear\_f;

line(x, y, x1, y1);

line(x1, y1, x2, y2);

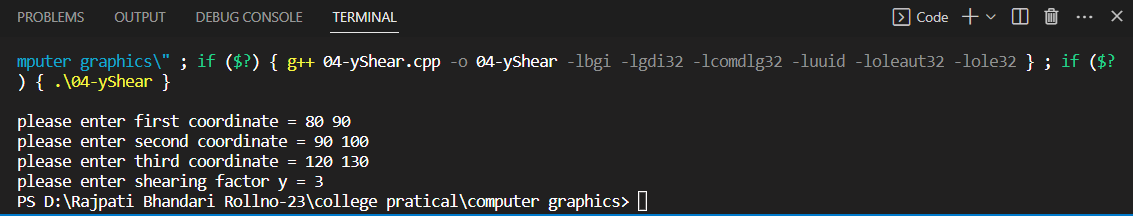
line(x2, y2, x, y);

getch();

closegraph();

return 0;

}

**Output:**

****