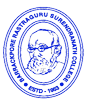
**Barrackpore Rastraguru Surendranath College**

**B.Sc. (Hons.) in Computer Science**

**Semester- VI**

****

**B.Sc. Computer Science Honours Semester-VI Examination 2023**

Name : LABANI DAS

College Roll No : 2001055

Semester : SEM-VI

WBSU Reg. No. : 1032021100062

Subject : Computer Science Hons. (CMSA)

Paper Code : CMSACOR14P

Paper Name : Computer Graphics

**INDEX**

|  |  |  |
| --- | --- | --- |
| **Program No.** | **Program Name** | **Page No.** |
| 1 | Implement DAA line drawing algorithm with end points (100,100), (250, 280). | 4 |
| 2 | Implemeent Bresenham line drawing algorithm with end points (100,100), (250, 280). | 5 |
| 3 | Write a Program in C language to implement the Bresenham circle drawing algorithm. Test your program to draw the circles, center (300,300), radius 75 units. | 7 |
| 4 | Write a Program in C language to implement the Bresenham circle drawing algorithm. Test your program to draw the circles, center (300,300), radius 100 units | 8 |
| 5 | Write a Program in C language to implement the Midpoint circle drawing algorithm. Test your program to draw the circles, center (300,300), radius 75 units. | 10 |
| 6 | Write a Program in C language to implement the Midpoint circle drawing algorithm. Test your program to draw the circles, center (300,300), radius 100 units. | 11 |
| 7 | Write a program in C language that will translate a triangle with coordinates (200,200), (100,300), & (300,300) with translation parameters Show translated | 13 |
| 8 | Write a program in C language that will translate a triangle with coordinates (200,200), (100,300), & (300,300) with translation parameters Show translated triangle. | 14 |
| 9 | Write a program in C language that will translate a triangle with coordinates (200,200), (100,300), & (300,300) with translation parameters Show translated triangle | 16 |
| 10 | Write a program in C language that will scale a triangle with coordinates (200,200), (100,300), & (300,300) with scaling parameters  Scale with respect to Origin | 18 |
| 11 | Write a program in C language that will scale a triangle with coordinates (200,200), (100,300), & (300,300) with scaling parameters  Scale with respect to Centroid. | 17 |
| 12 | Write a program in C language that will scale a triangle with coordinates (200,200), (100,300), & (300,300) with scaling parameters  Scale with respect to  Origin | 19 |
| 13 | Write a program in C language that will scale a triangle with coordinates (200,200), (100,300), & (300,300) with scaling parameters  Scale with respect to  Centroid | 20 |
| 14 | Rotate a triangle with coordinates (200,100), (100,300), (300,300) by 30 degrees with respect to the Origin | 22 |
| 15 | Rotate a triangle with coordinates (200,100), (100,300), (300,300) by 30 degrees with respect to the Centroid. | 24 |
| 16 | Consider a clipping window defined by the coordinates (100, 300) and (400, 100) and the lines with the following endpoints:   1. (150, 275), (300, 150) 2. (300, 350), (450, 350) 3. (200, 50), (500, 250)   Use Cohen Sutherland Clipping algorithm to display lines in different color and clip the part of lines which is outside the window | 26 |
| 17 | Write a program to draw hermite curve | 29 |
| 18 | Write a program to draw beizer curve | 30 |

**WRITE A PROGRAM IN C LANGUAGE TO :**

**PROGRAM NUMBER: 1**

Implement DAA line drawing algorithm with end points (100,100), (250, 280).

**Program code:**

// C program for DDA line generation

#include <graphics.h>

#include <math.h>

#include <stdio.h>

#include <cstdlib> // for using abs function

// DDA Function for line generation

void DDA(int X0, int Y0, int X1, int Y1)

{

    // calculate dx &dy

    int dx = X1 - X0;

    int dy = Y1 - Y0;

    // calculate steps required for generating pixels

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

    // calculate increment in x & y for each steps

    float Xinc = dx / (float)steps;

    float Yinc = dy / (float)steps;

    // Put pixel for each step

    float X = X0;

    float Y = Y0;

    for (int i = 0; i<= steps; i++) {

        putpixel(round(X), round(Y),

                WHITE); // put pixel at (X,Y)

        X += Xinc; // increment in x at each step

        Y += Yinc; // increment in y at each step

        delay(100); // for visualization of line-

                    // generation step by step

    }

}

int main()

{

    int gd = DETECT, gm;

    // Initialize graphics function

    initgraph(&gd, &gm, "");

    int X0 = 2, Y0 = 2, X1 = 14, Y1 = 16;

    // Function call

    DDA(100, 100, 250, 280);

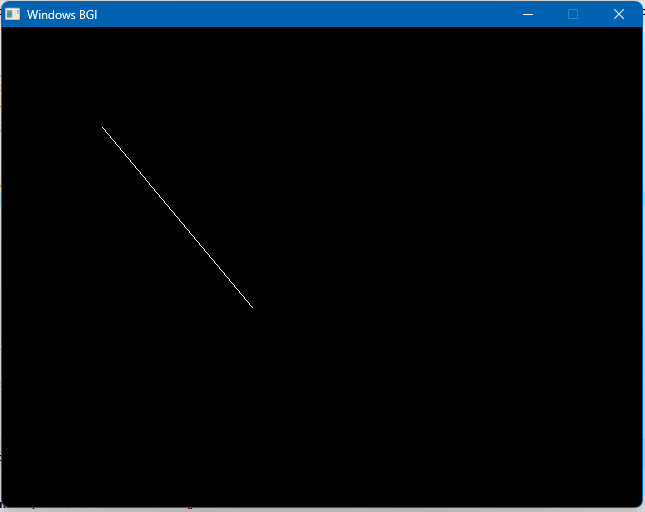
    getch();

    closegraph();

    return 0;

}

**Program output:**



**PROGRAM NUMBER: 2**

Implemeent Bresenham line drawing algorithm with end points (100,100), (250, 280).

**Program code:**

/\*program to draw line using Bresenham line drawing algorithm\*/

#include <stdio.h>

#include <graphics.h>

void drawline(int x0, int y0, int x1, int y1)

{

    int dx, dy, p, x, y;

    dx = x1 - x0;

    dy = y1 - y0;

    x = x0;

    y = y0;

    p = 2 \* dy - dx;

    while (x < x1)

    {

        if (p >= 0)

        {

            putpixel(x, y, 7);

            y = y + 1;

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

            delay(100);

        }

        else

        {

            putpixel(x, y, 7);

            p = p + 2 \* dy;

        }

        x = x + 1;

        delay(100);

    }

}

int main()

{

    int gdriver = DETECT, gmode, error, x0, y0, x1, y1;

    initgraph(&gdriver, &gmode, "");

    drawline(100, 100, 250, 280);

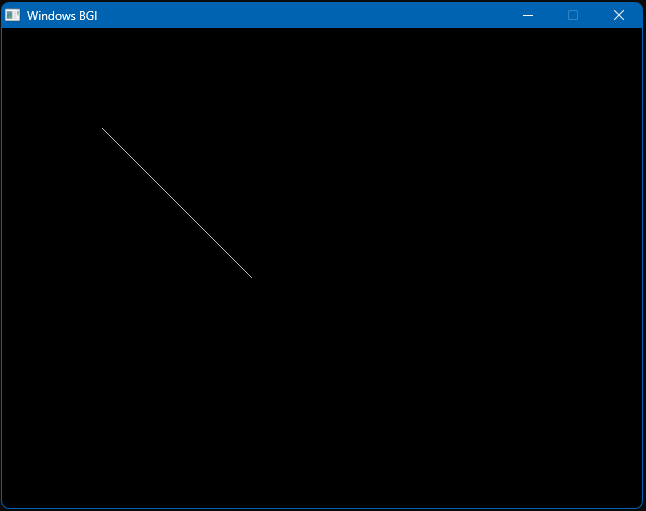
    getch();

    closegraph();

    return 0;

}

**Program output:**



**PROGRAM NUMBER: 3**

Write a Program in C language to implement the Bresenham circle drawing algorithm. Test your program to draw the circles, center (300,300), radius 75 units.

**Program code:**

#include <stdio.h>

#include <dos.h>

#include <graphics.h>

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

{

    putpixel(xc + x, yc + y, RED);

    putpixel(xc - x, yc + y, GREEN);

    putpixel(xc + x, yc - y, BLUE);

    putpixel(xc - x, yc - y, YELLOW);

    putpixel(xc + y, yc + x, CYAN);

    putpixel(xc - y, yc + x, RED);

    putpixel(xc + y, yc - x, YELLOW);

    putpixel(xc - y, yc - x, BROWN);

}

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

{

    int x = 0, y = r;

    int d = 3 - 2 \* r;

    drawCircle(xc, yc, x, y);

    while (y >= x)

    {

        x++;

        if (d > 0)

        {

            y--;

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

        }

        else

            d = d + 4 \* x + 6;

        drawCircle(xc, yc, x, y);

        delay(50);

    }

}

int main()

{

    int xc = 300, yc = 300, r = 75;

    int gd = DETECT, gm;

    initgraph(&gd, &gm, ""); // initialize graph

    circleBres(xc, yc, r);   // function call

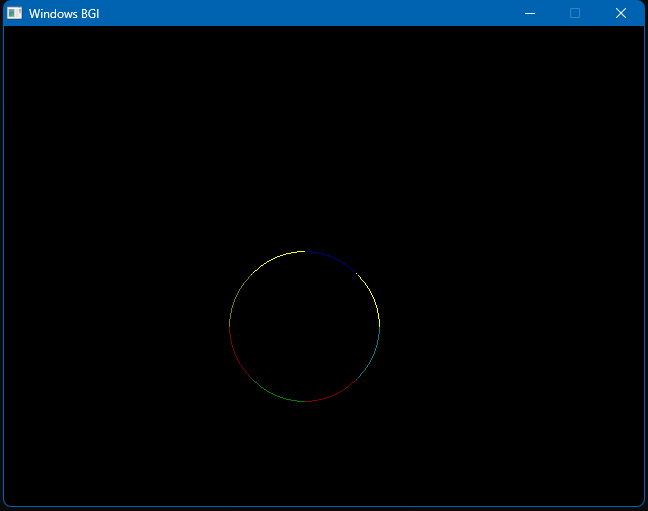
    getch();

    closegraph();

    return 0;

}

**Program output:**



**PROGRAM NUMBER: 4**

Write a Program in C language to implement the Bresenham circle drawing algorithm. Test your program to draw the circles, center (300,300), radius 100 units.

**Program code:**

/\*C-program for circle drawing using Bresenham's Algorithm in computer-graphics\*/

#include <stdio.h>

#include <dos.h>

#include <graphics.h>

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

{

    putpixel(xc + x, yc + y, RED);

    putpixel(xc - x, yc + y, GREEN);

    putpixel(xc + x, yc - y, BLUE);

    putpixel(xc - x, yc - y, YELLOW);

    putpixel(xc + y, yc + x, CYAN);

    putpixel(xc - y, yc + x, RED);

    putpixel(xc + y, yc - x, YELLOW);

    putpixel(xc - y, yc - x, BROWN);

}

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

{    int x = 0, y = r;

    int d = 3 - 2 \* r;

    drawCircle(xc, yc, x, y);

    while (y >= x)

    {

        x++;

        if (d > 0)        {            y--;

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

        else

            d = d + 4 \* x + 6;

        drawCircle(xc, yc, x, y);

        delay(50);

    }

}

int main(){

    int xc = 300, yc = 300, r = 100;

    int gd = DETECT, gm;

    initgraph(&gd, &gm, ""); // initialize graph

    circleBres(xc, yc, r);   // function call

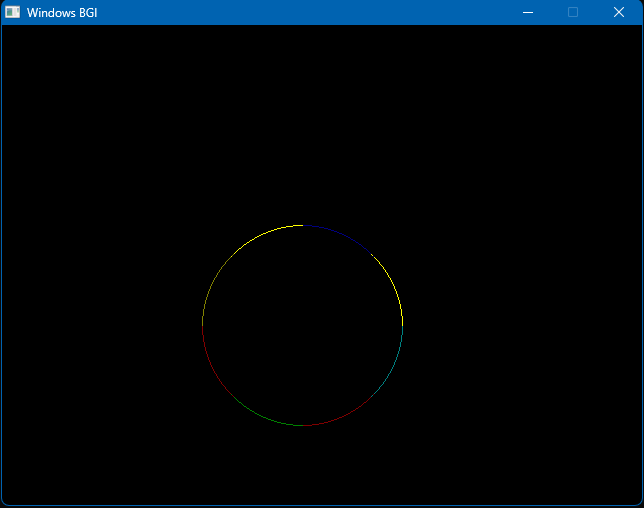
    getch();

    closegraph();

    return 0;

}

**Program output:**



**PROGRAM NUMBER: 5**

Write a Program in C language to implement the Midpoint circle drawing algorithm. Test your program to draw the circles, center (300,300), radius 75 units.

**Program code:**

#include <stdio.h>

#include <graphics.h>

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

{

    int x, y;

    float d;

    d = 1.25 - r;

    x = 0;

    y = r;

    do

    {

        if (d < 0)

        {

            x = x + 1;

            d = d + 2 \* x + 1;

        }

        else        {            x = x + 1;

            y = y - 1;

            d = d + 2 \* x - 2 \* y + 10;

            delay(100);

        }

        putpixel(xc + x, yc + y, 5);

        putpixel(xc - y, yc - x, 6);

        putpixel(xc + y, yc - x, 7);

        putpixel(xc - y, yc + x, 8);

        putpixel(xc + y, yc + x, 9);

        putpixel(xc - x, yc - y, 4);

        putpixel(xc + x, yc - y, 3);

        putpixel(xc - x, yc + y, 2);

    } while (x < y);

}

int main()

{

    int gd = DETECT, gm;

    int xc, yc, r;

    xc = 300;

    yc = 300;

    r = 75;

    initgraph(&gd, &gm, "");

    midp(r, xc, yc);

    delay(1500);

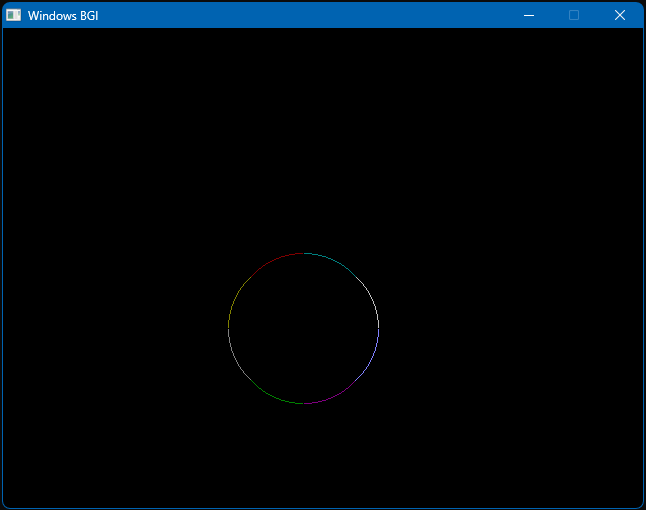
    getch();

    closegraph();

    return 0;

}

**Program output:**



**PROGRAM NUMBER: 6**

Write a Program in C language to implement the Midpoint circle drawing algorithm. Test your program to draw the circles, center (300,300), radius 100 units.

**Program code:**

#include <stdio.h>

#include <graphics.h>

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

{

    int x, y;

    float d;

    d = 1.25 - r;

    x = 0;

    y = r;

    do

    {

        if (d < 0)

        {

            x = x + 1;

            d = d + 2 \* x + 1;

        }

        else

        {

            x = x + 1;

            y = y - 1;

            d = d + 2 \* x - 2 \* y + 10;

            delay(100);

        }

        putpixel(xc + x, yc + y, 5);

        putpixel(xc - y, yc - x, 6);

        putpixel(xc + y, yc - x, 7);

        putpixel(xc - y, yc + x, 8);

        putpixel(xc + y, yc + x, 9);

        putpixel(xc - x, yc - y, 4);

        putpixel(xc + x, yc - y, 3);

        putpixel(xc - x, yc + y, 2);

    } while (x < y);

}

int main()

{

    int gd = DETECT, gm;

    int xc, yc, r;

    xc = 300;

    yc = 300;

    r = 100;

    initgraph(&gd, &gm, "");

    midp(r, xc, yc);

    delay(1500);

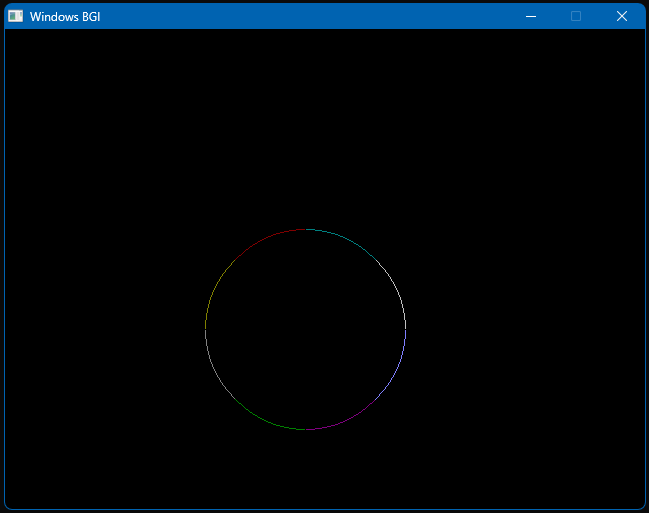
    getch();

    closegraph();

    return 0;

}

**Program output:**



**PROGRAM NUMBER: 7**

Write a program in C language that will translate a triangle with coordinates (200,200), (100,300), & (300,300) with translation parameters Show translated

**Program code:**

#include <conio.h>

#include <graphics.h>

#include <stdio.h>

int main()

{

    int gd = DETECT, gm;

    int x, y, x1, y1, x2, y2, tx, ty;

    x = 200;

    y = 200;

    x1 = 100;

    y1 = 300;

    x2 = 300;

    y2 = 300;

    initgraph(&gd, &gm, "");

    line(x, y, x1, y1);

    line(x1, y1, x2, y2);

    line(x2, y2, x, y);

    tx = 50;

    ty = 0;

    setcolor(RED);

    line(x + tx, y + ty, x1 + tx, y1 + ty);

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

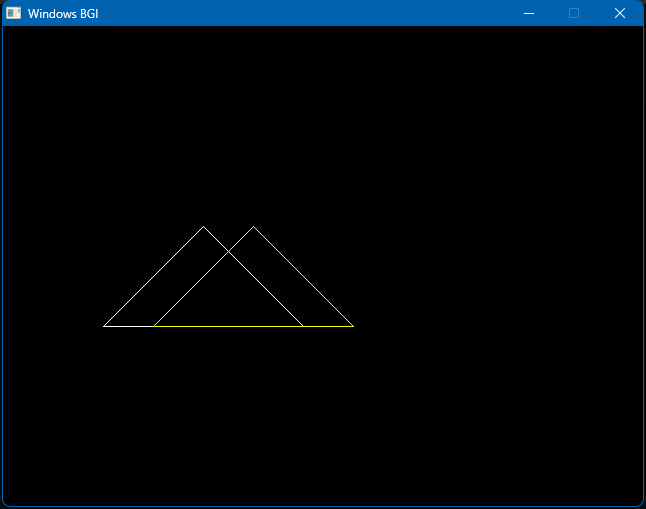
    line(x2 + tx, y2 + ty, x + tx, y + ty);

    getch();

    closegraph();

}

**Program output:**



**PROGRAM NUMBER: 8**

Write a program in C language that will translate a triangle with coordinates (200,200), (100,300), & (300,300) with translation parameters Show translated triangle.

**Program code:**

#include <conio.h>

#include <graphics.h>

#include <stdio.h>

int main()

{

    int gd = DETECT, gm;

    int x, y, x1, y1, x2, y2, tx, ty;

    x = 200;

    y = 200;

    x1 = 100;

    y1 = 300;

    x2 = 300;

    y2 = 300;

    initgraph(&gd, &gm, "");

    line(x, y, x1, y1);

    line(x1, y1, x2, y2);

    line(x2, y2, x, y);

    tx = 0;

    ty = 50;

    setcolor(RED);

    line(x + tx, y + ty, x1 + tx, y1 + ty);

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

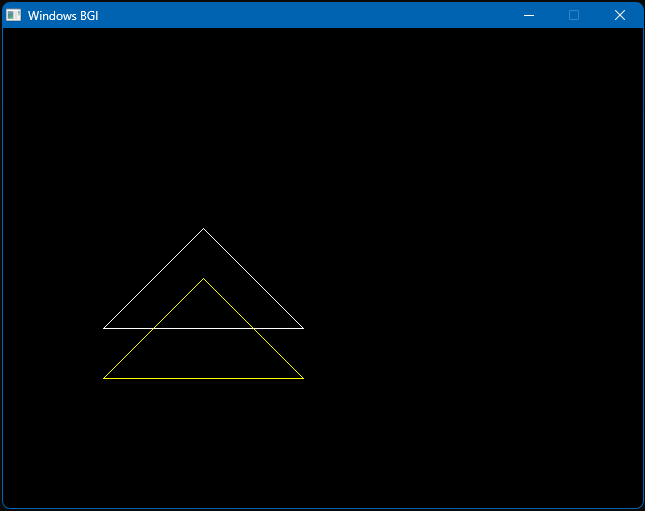
    line(x2 + tx, y2 + ty, x + tx, y + ty);

    getch();

    closegraph();

}

**Program output:**



**PROGRAM NUMBER: 9**

Write a program in C language that will translate a triangle with coordinates (200,200), (100,300), & (300,300) with translation parameters Show translated triangle.

**Program code:**

#include <conio.h>

#include <graphics.h>

#include <stdio.h>

int main(){

    int gd = DETECT, gm;

    int x, y, x1, y1, x2, y2, tx, ty;

    x = 200;

    y = 200;

    x1 = 100;

    y1 = 300;

    x2 = 300;

    y2 = 300;

    initgraph(&gd, &gm, "");

    line(x, y, x1, y1);

    line(x1, y1, x2, y2);

    line(x2, y2, x, y);

    tx = 50;

    ty = 50;

    setcolor(RED);

    line(x + tx, y + ty, x1 + tx, y1 + ty);

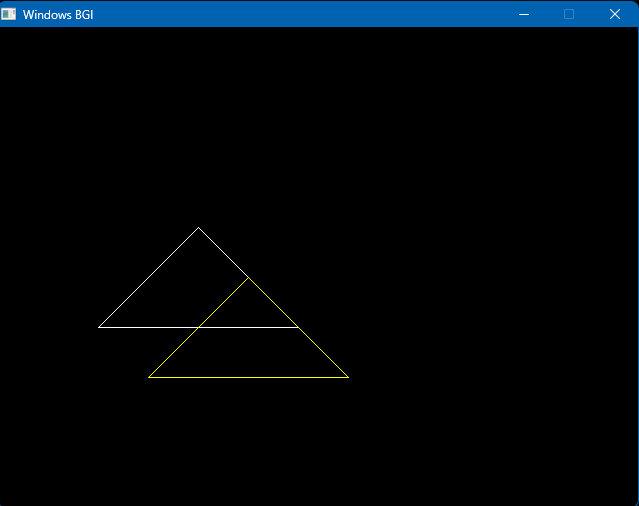
    line(x1 + tx, y1 + ty, x2 + tx, y2 + ty);

    line(x2 + tx, y2 + ty, x + tx, y + ty);

    getch();

    closegraph();}

**Program output:**



**PROGRAM NUMBER: 10**

Write a program in C language that will scale a triangle with coordinates (200,200), (100,300), & (300,300) with scaling parameters

Scale with respect to Origin

**Program code:**

#include <stdio.h>

#include <stdlib.h>

#include <graphics.h>

void scaleTriangle(int x1, int y1, int x2, int y2, int x3, int y3, float sx, float sy)

{

    int newX1, newY1, newX2, newY2, newX3, newY3;

    newX1 = x1 \* sx;

    newY1 = y1 \* sy;

    newX2 = x2 \* sx;

    newY2 = y2 \* sy;

    newX3 = x3 \* sx;

    newY3 = y3 \* sy;

    line(newX1, newY1, newX2, newY2);

    line(newX2, newY2, newX3, newY3);

    line(newX3, newY3, newX1, newY1);

}

int main()

{

    int gd, gm;

    gd = DETECT;

    initgraph(&gd, &gm, NULL);

    // Original triangle coordinates

    //    int x1 = 250, y1 = 100;

    //    int x2 = 300, y2 = 200;

    //    int x3 = 200, y3 = 200;

    int x1 = 200, y1 = 200;

    int x2 = 100, y2 = 300;

    int x3 = 300, y3 = 300;

    float sx = 2, sy = 2;

    // Draw the original triangle

    line(x1, y1, x2, y2);

    line(x2, y2, x3, y3);

    line(x3, y3, x1, y1);

    setcolor(YELLOW);

    // Scale the triangle

    scaleTriangle(x1, y1, x2, y2, x3, y3, sx, sy);

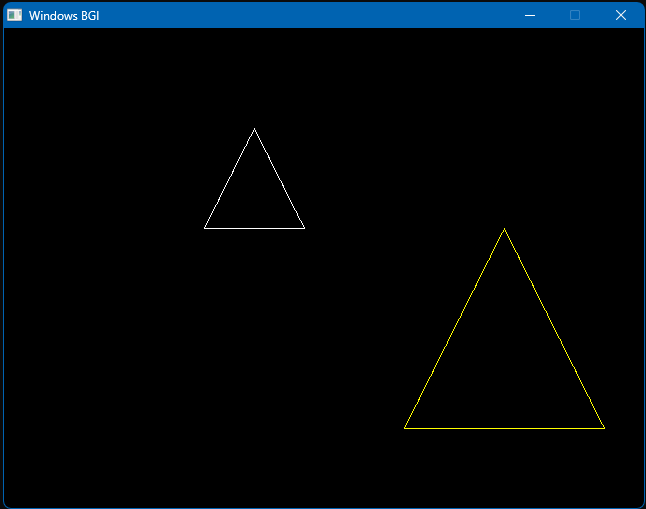
    getch();

    closegraph();

    return 0;

}

**Program output:**



**PROGRAM NUMBER: 11**

Write a program in C language that will scale a triangle with coordinates (200,200), (100,300), & (300,300) with scaling parameters

Scale with respect to Centroid.

**Program code:**

#include <graphics.h>

void scaleTriangle(int x1, int y1, int x2, int y2, int x3, int y3, float sx, float sy)

{

    int cx = (x1 + x2 + x3) / 3;

    int cy = (y1 + y2 + y3) / 3;

    // Calculate the scaled coordinates with respect to the centroid

    int nx1 = cx + (x1 - cx) \* sx;

    int ny1 = cy + (y1 - cy) \* sy;

    int nx2 = cx + (x2 - cx) \* sx;

    int ny2 = cy + (y2 - cy) \* sy;

    int nx3 = cx + (x3 - cx) \* sx;

    int ny3 = cy + (y3 - cy) \* sy;

    // Draw the scaled triangle

    line(nx1, ny1, nx2, ny2);

    line(nx2, ny2, nx3, ny3);

    line(nx3, ny3, nx1, ny1);

}

int main()

{

    int gd = DETECT, gm;

    initgraph(&gd, &gm, "");

    // Original triangle coordinates

    int x1 = 200, y1 = 200;

    int x2 = 100, y2 = 300;

    int x3 = 300, y3 = 300;

    // Scaling parameters

    float sx = 2;

    float sy = 2;

    // Draw the original triangle

    line(x1, y1, x2, y2);

    line(x2, y2, x3, y3);

    line(x3, y3, x1, y1);

    setcolor(YELLOW);

    // Scale the triangle and draw the scaled triangle

    scaleTriangle(x1, y1, x2, y2, x3, y3, sx, sy);

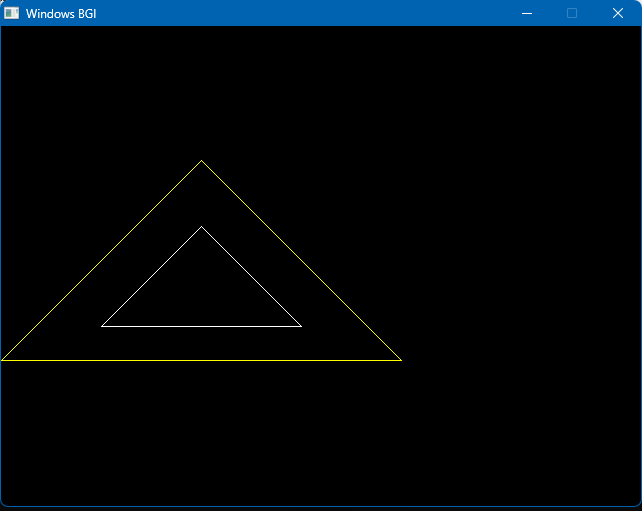
    getch();

    closegraph();

    return 0;

}

**Program output:**



**PROGRAM NUMBER: 12**

Write a program in C language that will scale a triangle with coordinates (200,200), (100,300), & (300,300) with scaling parameters

Scale with respect to  Origin

**Program code:**

#include <stdio.h>

#include <stdlib.h>

#include <graphics.h>

void scaleTriangle(int x1, int y1, int x2, int y2, int x3, int y3, float sx, float sy)

{

    int newX1, newY1, newX2, newY2, newX3, newY3;

    newX1 = x1 \* sx;

    newY1 = y1 \* sy;

    newX2 = x2 \* sx;

    newY2 = y2 \* sy;

    newX3 = x3 \* sx;

    newY3 = y3 \* sy;

    line(newX1, newY1, newX2, newY2);

    line(newX2, newY2, newX3, newY3);

    line(newX3, newY3, newX1, newY1);

}

int main()

{

    int gd, gm;

    gd = DETECT;

    initgraph(&gd, &gm, NULL);

    // Original triangle coordinates

    int x1 = 200, y1 = 200;

    int x2 = 100, y2 = 300;

    int x3 = 300, y3 = 300;

    // Scaling factors

    float sx = 0.5, sy = 0.5;

    // Draw the original triangle

    line(x1, y1, x2, y2);

    line(x2, y2, x3, y3);

    line(x3, y3, x1, y1);

    setcolor(CYAN);

    // Scale the triangle

    scaleTriangle(x1, y1, x2, y2, x3, y3, sx, sy);

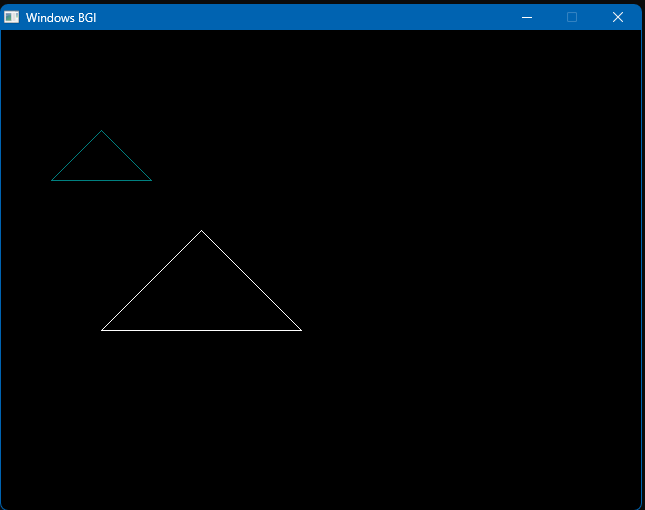
    getch();

    closegraph();

    return 0;

}

**Program output:**

****

**PROGRAM NUMBER: 13**

Write a program in C language that will scale a triangle with coordinates (200,200), (100,300), & (300,300) with scaling parameters

Scale with respect to  Centroid

**Program code:**

#include <graphics.h>

void scaleTriangle(int x1, int y1, int x2, int y2, int x3, int y3, float sx, float sy)

{

    int cx = (x1 + x2 + x3) / 3;

    int cy = (y1 + y2 + y3) / 3;

    // Calculate the scaled coordinates with respect to the centroid

    int nx1 = cx + (x1 - cx) \* sx;

    int ny1 = cy + (y1 - cy) \* sy;

    int nx2 = cx + (x2 - cx) \* sx;

    int ny2 = cy + (y2 - cy) \* sy;

    int nx3 = cx + (x3 - cx) \* sx;

    int ny3 = cy + (y3 - cy) \* sy;

    // Draw the scaled triangle

    line(nx1, ny1, nx2, ny2);

    line(nx2, ny2, nx3, ny3);

    line(nx3, ny3, nx1, ny1);

}

int main()

{

    int gd = DETECT, gm;

    initgraph(&gd, &gm, "");

    // Original triangle coordinates

    int x1 = 300, y1 = 100;

    int x2 = 400, y2 = 200;

    int x3 = 200, y3 = 200;

    // Scaling parameters

    float sx = 0.5;

    float sy = 0.5;

    // Draw the original triangle

    line(x1, y1, x2, y2);

    line(x2, y2, x3, y3);

    line(x3, y3, x1, y1);

    setcolor(CYAN);

    // Scale the triangle and draw the scaled triangle

    scaleTriangle(x1, y1, x2, y2, x3, y3, sx, sy);

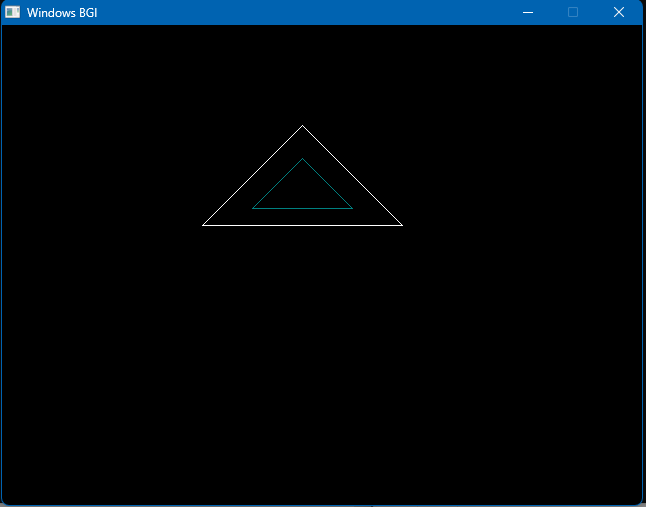
    getch();

    closegraph();

    return 0;

}

**Program output:**



**PROGRAM NUMBER: 14**

Rotate a triangle with coordinates (200,100), (100,300), (300,300) by 30 degrees with respect to the Origin

**Program code:**

#include <stdio.h>

#include <math.h>

#include <graphics.h>

// Function to rotate a point (x, y) by angle theta

void rotatePoint(int \*x, int \*y, double theta)

{

    double radian = (theta \* 3.14159) / 180.0;

    int tempX = \*x;

    int tempY = \*y;

    \*x = round(tempX \* cos(radian) - tempY \* sin(radian));

    \*y = round(tempX \* sin(radian) + tempY \* cos(radian));

}

// Function to rotate a triangle by angle theta

void rotateTriangle(int x1, int y1, int x2, int y2, int x3, int y3, double theta)

{

    // Rotate each vertex of the triangle

    rotatePoint(&x1, &y1, theta);

    rotatePoint(&x2, &y2, theta);

    rotatePoint(&x3, &y3, theta);

    // Draw the rotated triangle

    setcolor(YELLOW);

    line(x1, y1, x2, y2);

    line(x2, y2, x3, y3);

    line(x3, y3, x1, y1);

}

int main()

{

    int gd, gm;

    gd = DETECT;

    // Initialize graphics mode

    initgraph(&gd, &gm, NULL);

    // Set color to white

    setcolor(WHITE);

    // Define the coordinates of the triangle

    int x1 = 300, y1 = 100;

    int x2 = 400, y2 = 200;

    int x3 = 200, y3 = 200;

    // Draw the original triangle

    line(x1, y1, x2, y2);

    line(x2, y2, x3, y3);

    line(x3, y3, x1, y1);

    // Rotate the triangle by 30 degrees

    double theta = 30;

    rotateTriangle(x1, y1, x2, y2, x3, y3, theta);

    // Delay to see the result

    delay(5000);

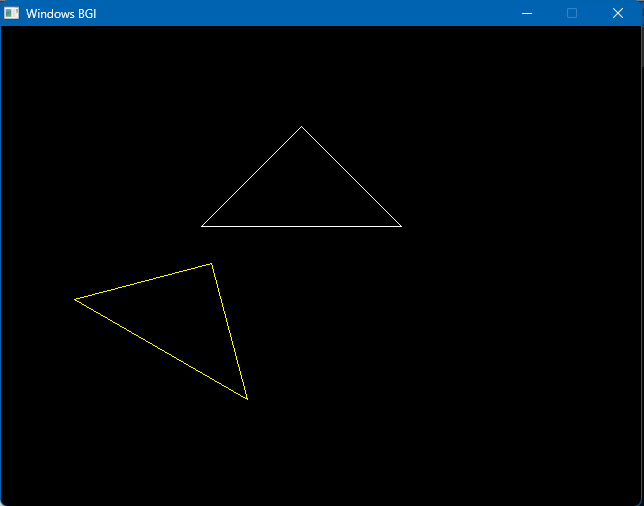
    // Close graphics mode

    closegraph();

    return 0;

}

**Program output:**



**PROGRAM NUMBER: 15**

Rotate a triangle with coordinates (200,100), (100,300), (300,300) by 30 degrees with respect to the Centroid.

**Program code:**

#include <stdio.h>

#include <stdlib.h>

#include <math.h>

#include <graphics.h>

void drawTriangle(int x1, int y1, int x2, int y2, int x3, int y3)

{

    line(x1, y1, x2, y2);

    line(x2, y2, x3, y3);

    line(x3, y3, x1, y1);

}

void rotateTriangle(int x1, int y1, int x2, int y2, int x3, int y3, float angle)

{

    // Calculate centroid

    int centroid\_x = (x1 + x2 + x3) / 3;

    int centroid\_y = (y1 + y2 + y3) / 3;

    // Convert angle from degrees to radians

    float radian = angle \* (M\_PI / 180.0);

    // Apply rotation transformation to each vertex

    int new\_x1 = centroid\_x + (x1 - centroid\_x) \* cos(radian) - (y1 - centroid\_y) \* sin(radian);

    int new\_y1 = centroid\_y + (x1 - centroid\_x) \* sin(radian) + (y1 - centroid\_y) \* cos(radian);

    int new\_x2 = centroid\_x + (x2 - centroid\_x) \* cos(radian) - (y2 - centroid\_y) \* sin(radian);

    int new\_y2 = centroid\_y + (x2 - centroid\_x) \* sin(radian) + (y2 - centroid\_y) \* cos(radian);

    int new\_x3 = centroid\_x + (x3 - centroid\_x) \* cos(radian) - (y3 - centroid\_y) \* sin(radian);

    int new\_y3 = centroid\_y + (x3 - centroid\_x) \* sin(radian) + (y3 - centroid\_y) \* cos(radian);

    // Draw the rotated triangle

    setcolor(CYAN);

    drawTriangle(new\_x1, new\_y1, new\_x2, new\_y2, new\_x3, new\_y3);

}

int main()

{

    int gd, gm;

    gd = DETECT;

    initgraph(&gd, &gm, "");

    // Coordinates of the original triangle

    int x1 = 200, y1 = 200;

    int x2 = 100, y2 = 300;

    int x3 = 300, y3 = 300;

    // Draw the original triangle

    drawTriangle(x1, y1, x2, y2, x3, y3);

    // Rotate the triangle by 30 degrees

    float angle = 30.0;

    rotateTriangle(x1, y1, x2, y2, x3, y3, angle);

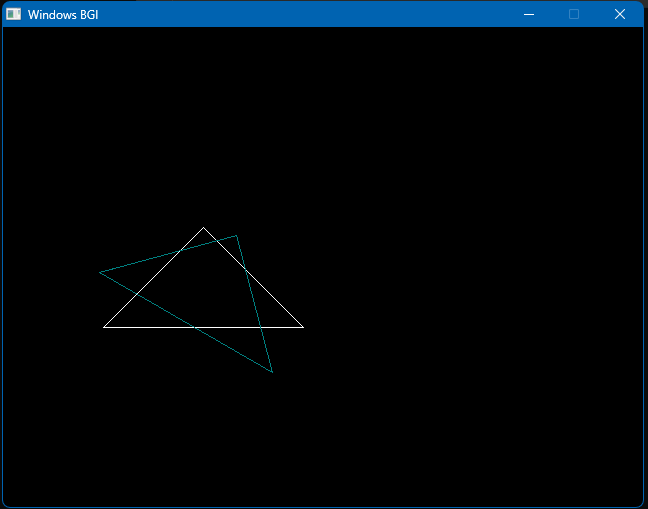
    delay(50000); // Delay for 5 seconds to see the result

    closegraph();

    return 0;

}

**Program output:**



**PROGRAM NUMBER: 16**

Consider a clipping window defined by the coordinates (100, 300) and (400, 100) and the lines with the following endpoints:

1. (150, 275), (300, 150)
2. (300, 350), (450, 350)
3. (200, 50), (500, 250)

Use Cohen Sutherland Clipping algorithm to display lines in different color and clip the part of lines which is outside the window

**Program code:**

#include <stdio.h>

#include <conio.h>

#include <graphics.h>

// Define the clipping window coordinates

int xmin = 100, ymin = 100;

int xmax = 400, ymax = 300;

// Define the region codes

#define INSIDE 0

#define LEFT 1

#define RIGHT 2

#define BOTTOM 4

#define TOP 8

// Calculate the region code for a given point

int calculateRegionCode(int x, int y)

{

    int code = INSIDE;

    if (x < xmin)

        code |= LEFT;

    else if (x > xmax)

        code |= RIGHT;

    if (y < ymin)

        code |= BOTTOM;

    else if (y > ymax)

        code |= TOP;

    return code;

}

// Clip the line using the Cohen-Sutherland algorithm

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

{

    int code1 = calculateRegionCode(x1, y1);

    int code2 = calculateRegionCode(x2, y2);

    int accept = 0;

    while (1)

    {

        if ((code1 == 0) && (code2 == 0))

        {

            accept = 1; // Line is completely inside the window

            break;

        }

        else if (code1 & code2)

        {

            break; // Line is completely outside the window

        }

        else

        {

            int x, y;

            int code = (code1 != 0) ? code1 : code2;

            if (code & TOP)

            {

                x = x1 + (x2 - x1) \* (ymax - y1) / (y2 - y1);

                y = ymax;

            }

            else if (code & BOTTOM)

            {

                x = x1 + (x2 - x1) \* (ymin - y1) / (y2 - y1);

                y = ymin;

            }

            else if (code & RIGHT)

            {

                y = y1 + (y2 - y1) \* (xmax - x1) / (x2 - x1);

                x = xmax;

            }

            else if (code & LEFT)

            {

                y = y1 + (y2 - y1) \* (xmin - x1) / (x2 - x1);

                x = xmin;

            }

            if (code == code1)

            {

                x1 = x;

                y1 = y;

                code1 = calculateRegionCode(x1, y1);

            }

            else

            {

                x2 = x;

                y2 = y;

                code2 = calculateRegionCode(x2, y2);

            }

        }

    }

    if (accept)

    {

        setcolor(YELLOW);

        line(x1, y1, x2, y2); // Display the clipped line in yellow

    }

}

int main()

{

    int gd = DETECT, gm;

    initgraph(&gd, &gm, "C:\\TC\\BGI"); // Update the path according to your setup

    setcolor(WHITE);

    rectangle(xmin, ymin, xmax, ymax); // Display the clipping window in white

    // Define the line endpoints

    int lines[][4] = {

        {150, 275, 300, 150}, // Line 1

        {300, 350, 450, 350}, // Line 2

        {200, 50, 500, 250}   // Line 3

    };

    // Display and clip each line

    for (int i = 0; i < sizeof(lines) / sizeof(lines[0]); i++)

    {

        int x1 = lines[i][0];

        int y1 = lines[i][1];

        int x2 = lines[i][2];

        int y2 = lines[i][3];

        setcolor(RED);

        line(x1, y1, x2, y2); // Display the original line in red

        cohenSutherlandClip(x1, y1, x2, y2); // Clip the line

    }

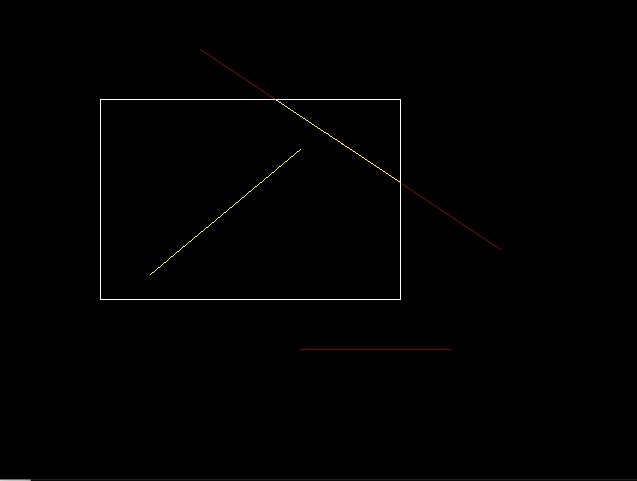
    getch();

    closegraph();

    return 0;

}

**Program output:**



**PROGRAM NUMBER: 17**

Write a program to draw hermite curve

**Program code:**

#include <stdio.h>

#include <conio.h>

#include <graphics.h>

// Hermite curve function

int hermite(int p0, int p1, int r0, int r1, double t)

{

    double t2 = t \* t;

    double t3 = t2 \* t;

    // Hermite basis functions

    double h1 = 2 \* t3 - 3 \* t2 + 1;

    double h2 = -2 \* t3 + 3 \* t2;

    double h3 = t3 - 2 \* t2 + t;

    double h4 = t3 - t2;

    // Calculate and return the point on the Hermite curve

    return (int)(h1 \* p0 + h2 \* p1 + h3 \* r0 + h4 \* r1);

}

int main()

{

    int gd = DETECT, gm;

    initgraph(&gd, &gm, "C:\\TC\\BGI"); // Update the path according to your setup

    int p0x = 100, p0y = 100; // Start point

    int p1x = 400, p1y = 400; // End point

    int r0x = 200, r0y = 200; // Start tangent

    int r1x = 300, r1y = 100; // End tangent

    double t;

    int x, y;

    setcolor(WHITE);

    line(p0x, p0y, p1x, p1y); // Draw the control line in white

    setcolor(YELLOW);

    for (t = 0; t <= 1; t += 0.01)

    {

        x = hermite(p0x, p1x, r0x, r1x, t);

        y = hermite(p0y, p1y, r0y, r1y, t);

        putpixel(x, y, YELLOW);

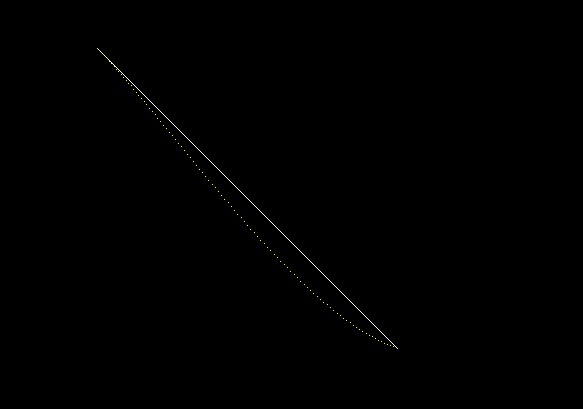
    }

    getch();

    closegraph();

    return 0;}

**Program output:**

****

**PROGRAM NUMBER: 18**

Write a program to draw beizer curve

**Program code:**

#include <stdio.h>

#include <conio.h>

#include <graphics.h>

// Bezier curve function

int bezier(int p0, int p1, int p2, int p3, double t)

{

    double t2 = t \* t;

    double t3 = t2 \* t;

    // Bezier basis functions

    double b1 = -t3 + 3 \* t2 - 3 \* t + 1;

    double b2 = 3 \* t3 - 6 \* t2 + 3 \* t;

    double b3 = -3 \* t3 + 3 \* t2;

    double b4 = t3;

    // Calculate and return the point on the Bezier curve

    return (int)(b1 \* p0 + b2 \* p1 + b3 \* p2 + b4 \* p3);

}

int main(){

    int gd = DETECT, gm;

    initgraph(&gd, &gm, "C:\\TC\\BGI"); // Update the path according to your setup

    int p0x = 100, p0y = 100; // Start point

    int p1x = 150, p1y = 300; // Control point 1

    int p2x = 300, p2y = 50;  // Control point 2

    int p3x = 400, p3y = 200; // End point

    double t;

    int x, y;

    setcolor(WHITE);

    line(p0x, p0y, p1x, p1y); // Draw the control lines in white

    line(p2x, p2y, p3x, p3y);

    setcolor(YELLOW);

    for (t = 0; t <= 1; t += 0.01)

    {

        x = bezier(p0x, p1x, p2x, p3x, t);

        y = bezier(p0y, p1y, p2y, p3y, t);

        putpixel(x, y, YELLOW);

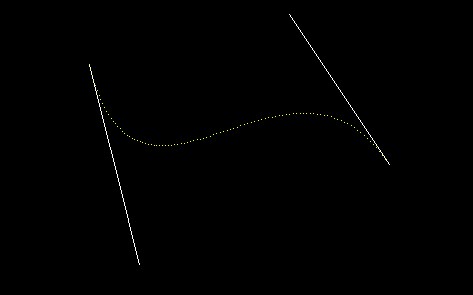
    }

    getch();

    closegraph();

    return 0;}

**Program output:**

****