**NAME:- SURWADE TRISHARAN RAJESH**

**ROLL NO:- 47**

**CLASS -MSC(CS)-IInd Year**

**3. Ellipse drawing algorithm**

#include <graphics.h>

#include <iostream.h>

#include <conio.h>

#include <dos.h> // For delay function

class Circle {

private:

int xc, yc, r; // Center and radius of the circle

public:

// Constructor to initialize the circle's center and radius

Circle(int centerX, int centerY, int radius) {

xc = centerX;

yc = centerY;

r = radius;

}

// Function to draw the circle using Bresenham's Midpoint Circle Algorithm

void draw() {

int x = 0;

int y = r;

int p = 1 - r; // Initial decision parameter

// Plot the initial points in all 8 octants

plotPoints(x, y);

// Continue plotting until x >= y

while (x < y) {

x++;

// Update decision parameter

if (p < 0) {

p = p + 2 \* x + 1; // Decision is to move horizontally

} else {

y--; // Decision is to move diagonally

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

}

// Plot the points in all 8 octants

plotPoints(x, y);

delay(50); // Delay for 50 milliseconds to make drawing visible

}

}

private:

// Function to plot the points in all 8 octants of the circle

void plotPoints(int x, int y) {

putpixel(xc + x, yc - y, WHITE); // Top right

putpixel(xc - x, yc - y, WHITE); // Top left

putpixel(xc + x, yc + y, WHITE); // Bottom right

putpixel(xc - x, yc + y, WHITE); // Bottom left

putpixel(xc + y, yc - x, WHITE); // Right top

putpixel(xc - y, yc - x, WHITE); // Left top

putpixel(xc + y, yc + x, WHITE); // Right bottom

putpixel(xc - y, yc + x, WHITE); // Left bottom

}

};

int main() {

int gd = DETECT, gm;

int xc, yc, r;

// Initialize graphics mode

initgraph(&gd, &gm, "C:\\TurboC3\\BGI"); // Specify the correct path to BGI folder

// Input center and radius of the circle

cout << "Enter the center coordinates (xc, yc) of the circle: ";

cin >> xc >> yc;

cout << "Enter the radius of the circle: ";

cin >> r;

// Create an object of Circle class and draw the circle

Circle circle(xc, yc, r);

circle.draw(); // Draw the circle using Midpoint Circle Algorithm

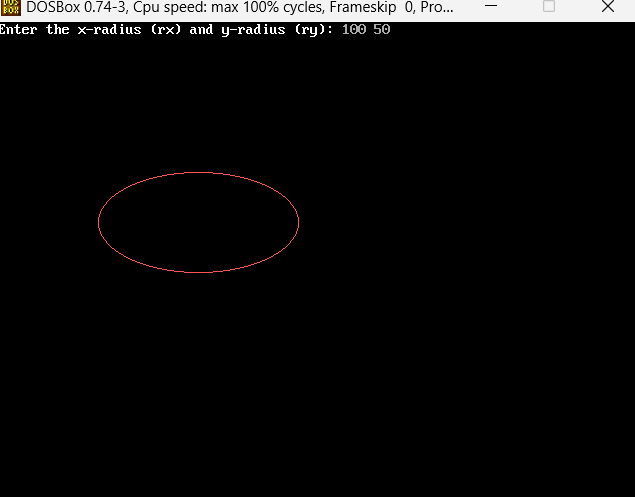
getch(); // Wait for user input

closegraph(); // Close graphics mode before exiting

return 0;

}

**OUTPUT:**

****