Problem Statement

Write a program that uses the Midpoint Circle Drawing Algorithm to draw a circle in a graphics window. The user should provide the center coordinates and the radius of the circle as input. The program must compute all points on the circle using the Midpoint Circle Drawing Algorithm and display them using the graphics library.

Background Theory

The Midpoint Circle Drawing Algorithm is a fundamental algorithm in computer graphics used to draw circles using only integer arithmetic, making it both efficient and suitable for real-time rendering systems. Unlike traditional methods that rely on floating-point calculations and trigonometric functions to determine the positions of circle points, this algorithm simplifies the process through the use of a decision parameter.

The core idea behind the algorithm is based on the symmetry of the circle. Since a circle is symmetric about its center, it is sufficient to compute the points for one-eighth (an octant) of the circle and reflect those points into the remaining seven octants. This significantly reduces the computational workload.

Algorithm

Step 1: Start

Step 2: Declare variables:

i. xc, yc as center coordinates of the circle

ii. r as radius of the circle

iii. x, y as current pixel position

iv. p as decision parameter

Step 3: Read values of xc, yc, r

Step 4: Initialize the first point on the circumference

x = 0

y = r

Step 5: Calculate initial decision parameter

p = 1 - r // (approximation of 5/4 - r for integer arithmetic)

Step 6: Repeat while x < y:

If p < 0:

x = x + 1

// y remains the same

p = p + 2*x + 1

Else:

x = x + 1

y = y - 1

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

Go to Step 7

Step 7: For each (x, y), determine the 8 symmetric points using:

(xc + x, yc + y)

(xc - x, yc + y)

(xc + x, yc - y)

(xc - x, yc - y)

(xc + y, yc + x)

(xc - y, yc + x)

(xc + y, yc - x)

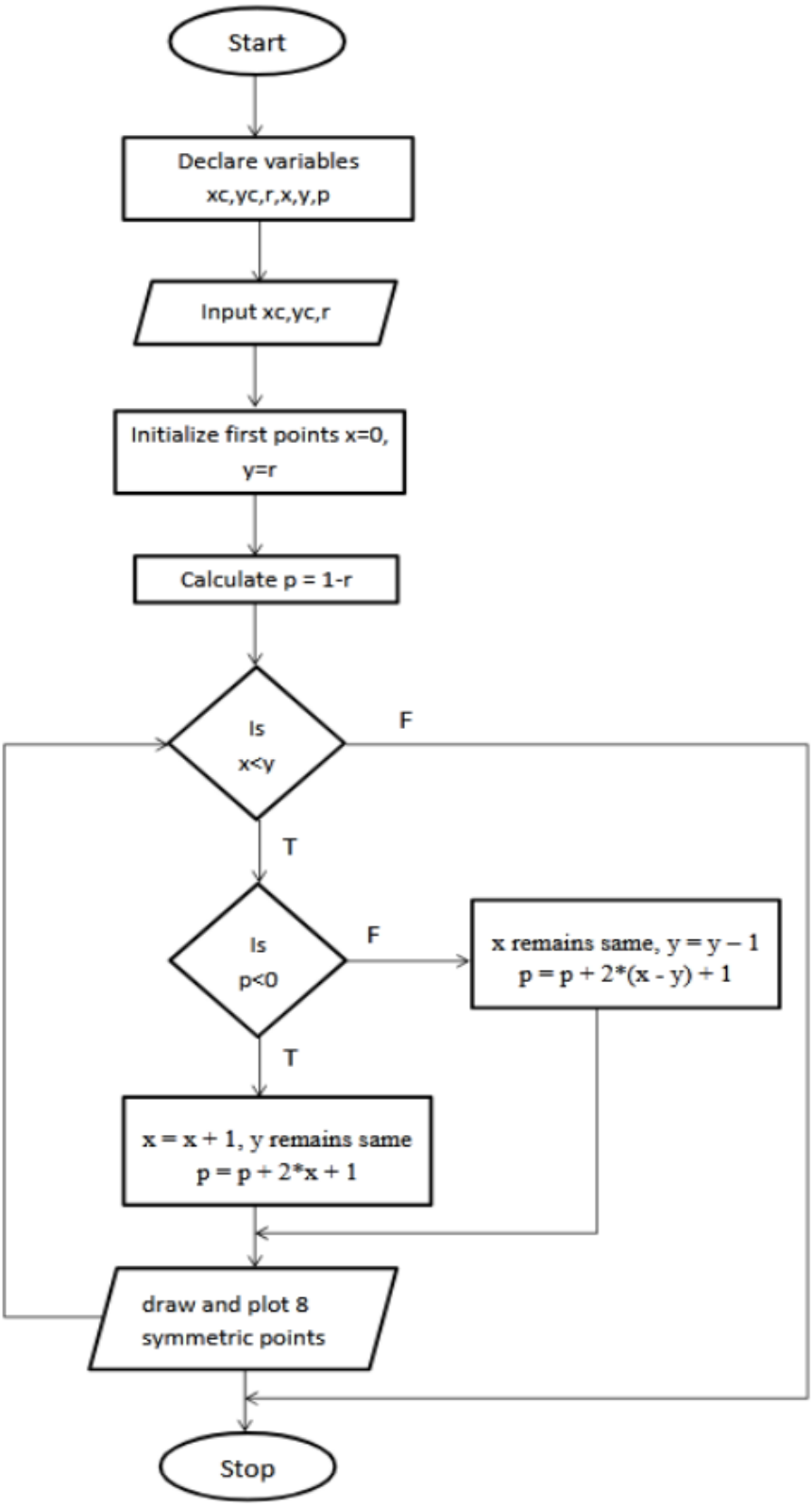
(xc - y, yc - x)

Step 8: Plot all 8 symmetric points

Step 9: Repeat steps 6 to 8 until $x \geq y$

Step 10: Stop

Flowchart


Start
Declare variables
xc,yc,r,x,y,p
Input xc,yc,r
Initialize first points x=0,
y=r
Calculate p = 1-r
Is
x<y
F
T
Is
p<0
F
x remains same, y = y – 1
p = p + 2*(x - y) + 1
T
x = x + 1, y remains same
p = p + 2*x + 1
draw and plot 8
symmetric points
Stop

Source code

```
#include <graphics.h>
#include <iostream>
using namespace std;

// Function to plot all 8 symmetric points
void plotCirclePoints(int xc, int yc, int x, int y) {
    putpixel(xc + x, yc + y, WHITE);
    putpixel(xc - x, yc + y, WHITE);
    putpixel(xc + x, yc - y, WHITE);
    putpixel(xc - x, yc - y, WHITE);
    putpixel(xc + y, yc + x, WHITE);
    putpixel(xc - y, yc + x, WHITE);
    putpixel(xc + y, yc - x, WHITE);
    putpixel(xc - y, yc - x, WHITE);
}

void midpointCircleAlgorithm(int xc, int yc, int r) {
    int x = 0;
    int y = r;
```

```
    // p0 = (5/4) - r ≈ 1 - r (as integer arithmetic)
    int p = 1 - r;

    plotCirclePoints(xc, yc, x, y);

    while (x < y) {
        x++;

        if (p < 0) {
            // Midpoint is inside the circle
            p = p + 2 * x + 1;
        } else {
            // Midpoint is outside or on the circle
            y--;
            p = p + 2 * (x - y) + 1;
        }

        plotCirclePoints(xc, yc, x, y);
    }
}
```

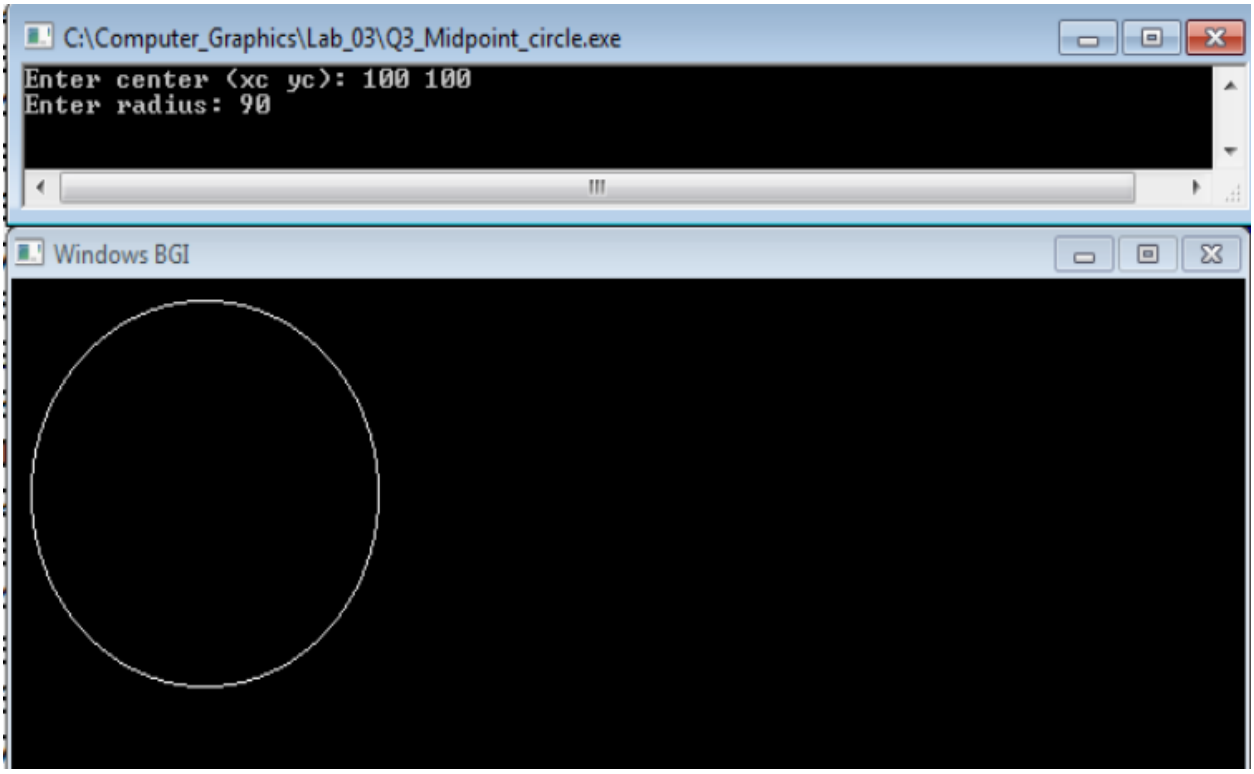
```
int main() {
    int gd = DETECT, gm;
    initgraph(&gd, &gm, "");

    int xc, yc, r;
    cout << "Enter center (xc yc): ";
    cin >> xc >> yc;
    cout << "Enter radius: ";
    cin >> r;

    midpointCircleAlgorithm(xc, yc, r);

    getch();
    closegraph();
    return 0;
}
```

Sample input and output



Conclusion:

The Midpoint Circle Drawing Algorithm offers an efficient and precise method for rendering circles in computer graphics using only integer arithmetic. This lab effectively demonstrated the power of the Midpoint Circle Algorithm through its implementation in C++ with the graphics.h library. It showcased how mathematical principles and algorithmic optimization can be applied to draw smooth, symmetric circles accurately and efficiently. The practical application reinforced the importance of discrete mathematics and algorithm design in the field of computer graphics.