Problem Statement

WAP that uses Bresenham's Line Drawing Algorithm to draw a straight line between two user-defined points, in a graphics window.

Background theory

The Bresenham line drawing algorithm provides an efficient way to render straight lines in computer graphics using only integer calculations. Unlike DDA, it eliminates the need for floating-point arithmetic, making it faster and more suitable for real-time rendering. Bresenham's method accurately determines the nearest pixel to the ideal line path by using a decision variable, ensuring crisp and consistent lines even with steep or shallow slopes. This lab demonstrated the power of algorithmic optimization and how discrete mathematics can be leveraged to draw precise lines efficiently in C++ graphics programming.

Algorithm

1. Input: Two points (x1, y1) and (x2, y2).
2. Calculate dx = |x2 - x1|, dy = |y2 - y1|.
3. Determine step directions:

   xinc = (x2 > x1) ? 1 : -1

   yinc = (y2 > y1) ? 1 : -1
4. Case 1 (|m| ≤ 1, dx ≥ dy):
   - Initialize p = 2*dy - dx.
   - For i = 0 to dx:
     i. Plot (x, y).
     ii. If p < 0:

         x += xinc, p += 2*dy.

iii. Else:

x += xinc, y += yinc, p += 2*(dy - dx).

5. Case 2 (|m| > 1, dy > dx):

- Initialize p = 2*dx - dy.
- For i = 0 to dy:

i. Plot (x, y).

ii. If p < 0:

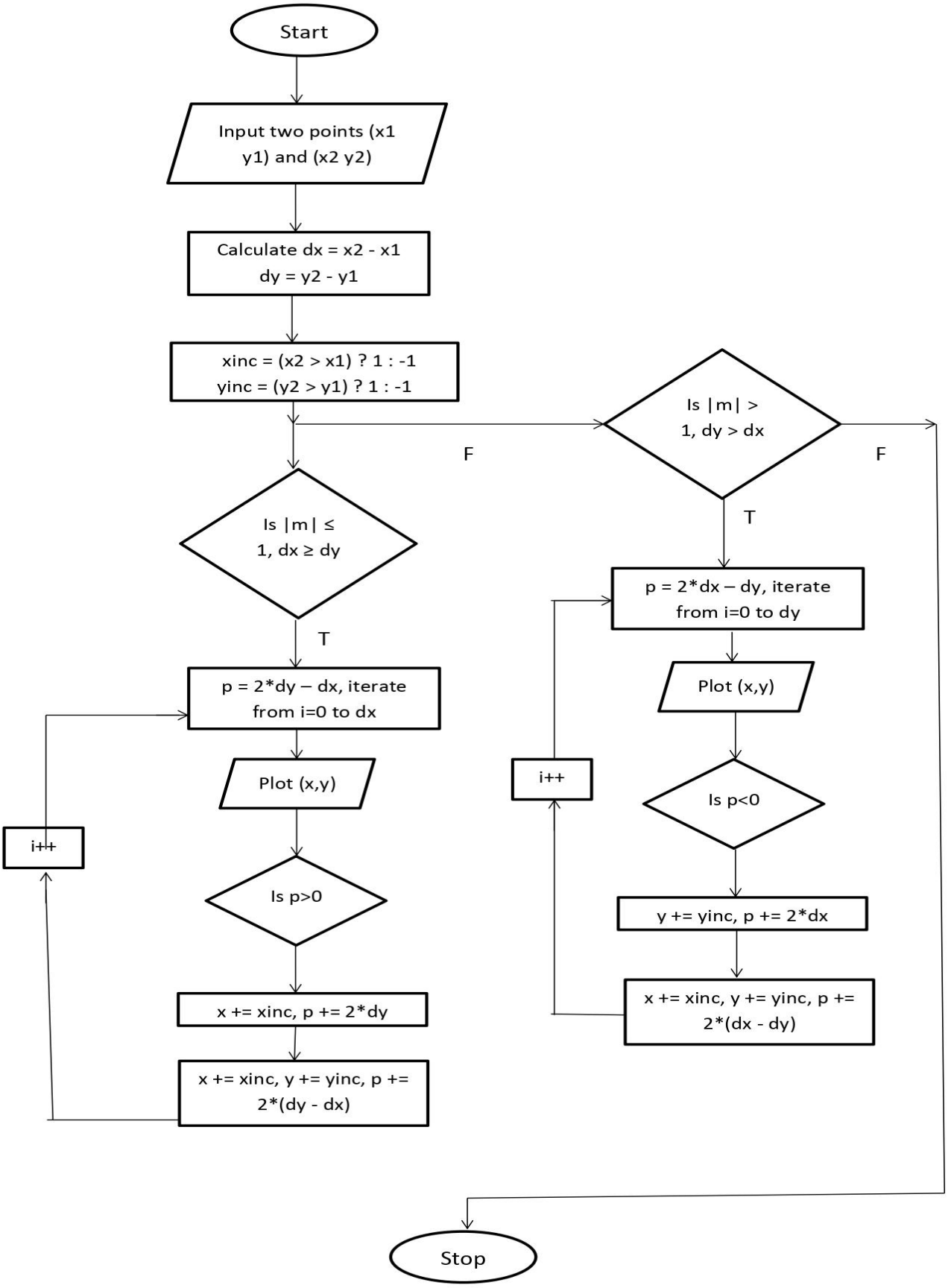
y += yinc, p += 2*dx.

iii. Else:

x += xinc, y += yinc, p += 2*(dx - dy)

6. Add delay to make clear visibility.
7. Stop.

Flowchart


Start
Input two points (x1 y1) and (x2 y2)
Calculate dx = x2 - x1 dy = y2 - y1
xinc = (x2 > x1) ? 1 : -1 yinc = (y2 > y1) ? 1 : -1
Is |m| ≤ 1, dx ≥ dy
T
F
p = 2*dy – dx, iterate from i=0 to dx
Plot (x,y)
Is p>0
x += xinc, p += 2*dy
x += xinc, y += yinc, p += 2*(dy - dx)
i++
Is |m| > 1, dy > dx
T
F
p = 2*dx – dy, iterate from i=0 to dy
Plot (x,y)
Is p<0
y += yinc, p += 2*dx
x += xinc, y += yinc, p += 2*(dx - dy)
i++
Stop

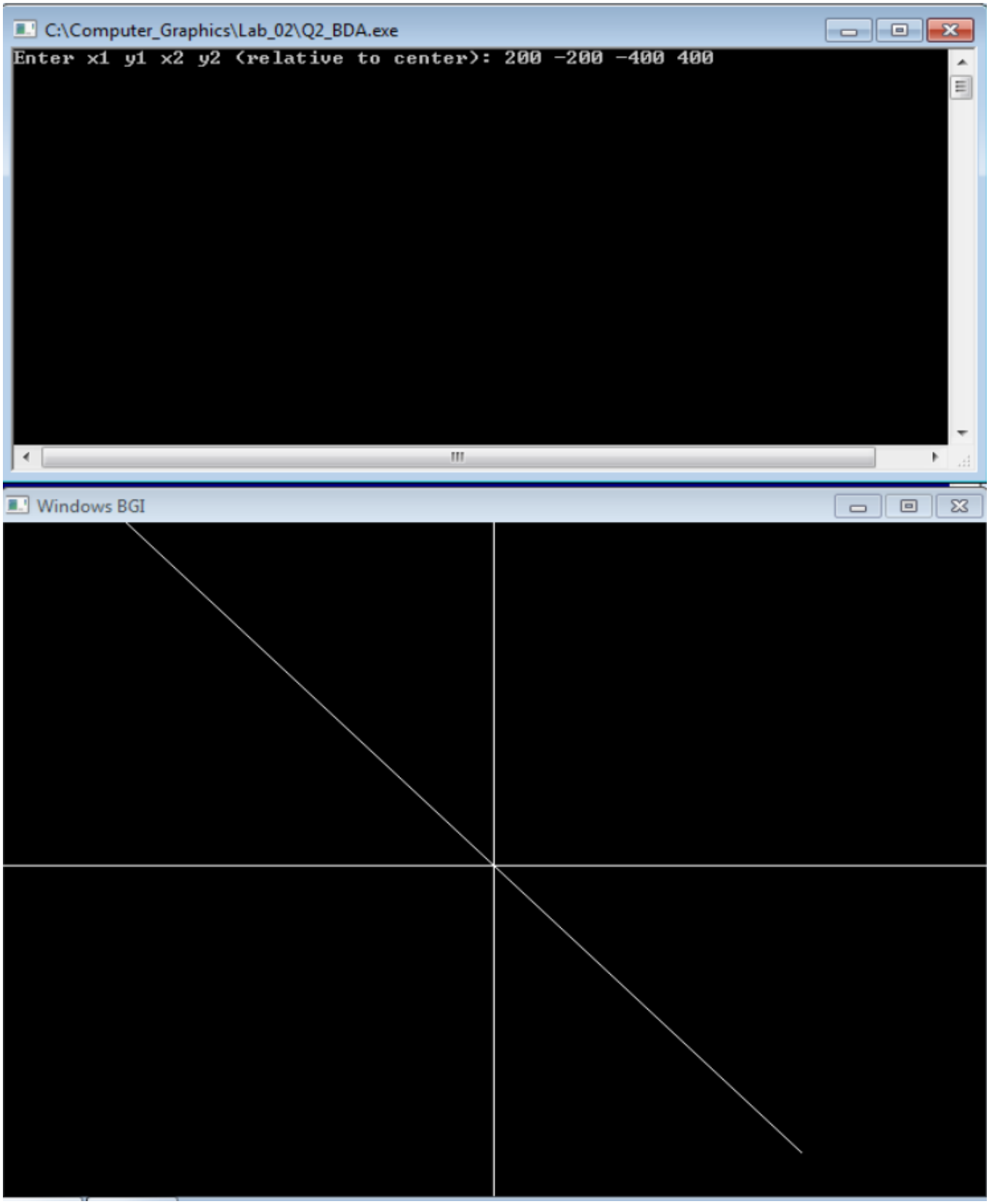
Source code

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

void bda(int x1, int y1, int x2, int y2) {
    int maxWidth = getmaxx();
    int maxHeight = getmaxy();
    int centerX = maxWidth / 2;
    int centerY = maxHeight / 2;

    line(0, centerY, maxWidth, centerY);
    line(centerX, 0, centerX, maxHeight);
    int dx = abs(x2 - x1);
    int dy = abs(y2 - y1);
    int x = x1;
    int y = y1;

    // Direction of steps
    int xinc = (x2 > x1) ? 1 : -1;
    int yinc = (y2 > y1) ? 1 : -1;
    int p;
```

```
if (dx >= dy) {
    p = 2 * dy - dx;
    for (int i = 0; i <= dx; i++) {
        putpixel(centerX + x, centerY - y, WHITE);
        if (p < 0) {
            x += xinc;
            p += 2 * dy;
        } else {
            x += xinc;
            y += yinc;
            p += 2 * dy - 2 * dx;
        }
        delay(5);
    }
}
else {
    p = 2 * dx - dy;
    for (int i = 0; i <= dy; i++) {
        putpixel(centerX + x, centerY - y, WHITE);
        if (p < 0) {
            y += yinc;
            p += 2 * dx;
```

```
            } else {
                x += xinc;
                y += yinc;
                p += 2 * dx - 2 * dy;
            }
            delay(5);
        }
    }
}
int main() {
    int gd = DETECT, gm;
    initgraph(&gd, &gm, NULL);
    int x1, y1, x2, y2;
    cout << "Enter x1 y1 x2 y2 (relative to center): ";
    cin >> x1 >> y1 >> x2 >> y2;
    bda(x1, y1, x2, y2);
    getch();
    closegraph();
    return 0;
}
```

Sample input and output



Conclusion

The Bresenham's Line Algorithm (BLA) provides an efficient and integer-only method for drawing lines in computer graphics. Unlike DDA, it avoids floating-point calculations, making it faster and more suitable for hardware implementation. By using a decision parameter (p), BLA optimally selects the next pixel, ensuring smooth and accurate line rendering for all slopes.

This lab demonstrated how mathematical optimization can enhance graphical computations, reinforcing the importance of algorithmic efficiency in real-time graphics programming. The implementation in C++ using graphics.h effectively illustrated BLA’s superiority over DDA in terms of speed and precision.