## **LAB 2**

## OBJECTIVE: To implement and understand the Bresenham’s Line Drawing Algorithm in C++ using Turbo C++ and graphics library functions.

**THEORY:**

The Bresenham’s Line Drawing Algorithm is an efficient method used in computer graphics to draw a straight line between two points. It uses only integer calculations, which makes it faster than the DDA algorithm, especially on systems where floating-point operations are expensive.

Key Concepts:

* Works using decision parameters to determine the next pixel position.
* Uses only integer addition, subtraction, and multiplication.
* Ideal for raster devices (like monitors) where pixel locations are integers.

**ALGORITHM:**

1. Input the coordinates of the two endpoints (x1, y1) and (x2, y2).
2. Calculate dx = x2 - x1 and dy = y2 - y1.
3. Initialize the decision parameter m = 2\*dy - dx.
4. Start at the first point (x1, y1).
5. For each step until x = x2:

* Plot the pixel using putpixel(x, y).
* Update the decision parameter m:
* If m < 0: increment x and update m += 2\*dy.
* Else: increment both x and y, and update m += 2\*dy - 2\*dx.

## PROGRAMS

*#include <graphics.h>*

*#include <iostream.h>*

*#include <conio.h>*

*int main() {*

*int x, y, x1, y1, x2, y2, dx, dy, m, i;*

*int gd = DETECT, gm;*

*initgraph(&gd, &gm,"c:\\turboc3\\bgi");*

*cout << "Enter first point (x1 y1): ";*

*cin >> x1 >> y1;*

*cout << "Enter second point (x2 y2): ";*

*cin >> x2 >> y2;*

*dx = x2 - x1;*

*dy = y2 - y1;*

*m = 2 \* dy - dx;*

*x = x1;*

*y = y1;*

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

*putpixel(x, y, WHITE);*

*if (m < 0) {*

*x = x + 1;*

*m = m + 2 \* dy;*

*} else {*

*x = x + 1;*

*y = y + 1;*

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

*}*

*}*

*getch();*

*closegraph();*

*return 0;*

*}*

Output:



RESULTS:

The program successfully implements Bresenham’s Line Drawing Algorithm and draws a line between two points using only integer calculations.

CONCLUSION:

The experiment demonstrates the working of Bresenham’s algorithm. It is more efficient than the DDA algorithm for line drawing, especially in terms of performance since it avoids floating-point arithmetic.