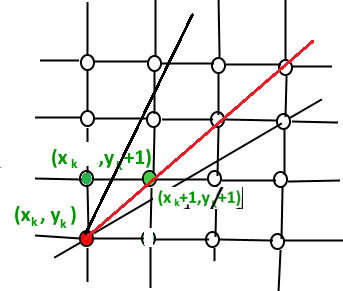
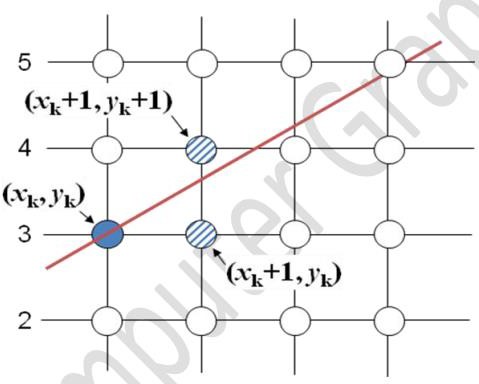
## Lab2

**OBJECTIVE:** TO IMPLEMENT BRESENHAM’S LINE DRAWING ALGORITHM FOR DRAWING A LINE SEGMENT BETWEEN TWO GIVEN ENDPOINTS A (x1, y1) AND B(x2, y2).

**THEORY:** This algorithm is used for scan converting a line. It was developed by Bresenham. It is an efficient method because it involves only integer addition, subtractions, and multiplication operations. These operations can be performed very rapidly so lines can be generated quickly.

In this method, next pixel selected is that one who has the least distance from true line.

* Move across the x axis in unit intervals and at each step choose between two different y coordinates when slope m<1.
* For example, from position (2, 3) we have to choose between (3, 3) and (3, 4). We would like the point that is closer to the original line.
* Move across the y axis in unit intervals and at each step choose between two different x coordinates when slope is m>=1.
* So we have to take decision to choose next point. So next pixels are selected based on the value of decision parameter p. The equations are given in below algorithm.

# BRESENHAM’S LINE DRAWING ALGORITHM

**Step 1:** Start

**Step 2:** Input starting point P1(x1, y1) and ending point P2 (x2,y2)

**Step 3:** Calculate the slope (m) of the required Line. **Step 4:** Identify the value of slope (m). m = dy/dx **Step 4.1:** If slope (m) is Less than 1 i.e.: m < 1

* **Step 4.1.1** : Calculate the constants dx, dy,, and (2dy – 2dx) and get the first value for the decision parameter as -
* p0 = 2dy − dx
* **Step 4.1.2** : At each Xk along the line, starting at k = 0, perform the following test −
* If pk < 0, the next point to plot is (xk + 1,yk) and pk+1 = pk + 2dy

else

* plot (xk+1,yk + 1)
* pk+1 = pk + 2dy − 2dx
* **Step 4.1.3** : Repeat until x2>=x1..

**Step 4.2** : If slope(m) is greater than or equal to 1 i.e: m >= 1

* + **Step 4.2.1** : Calculate the constants dx, dy, and (2dx – 2dy) and get the first value for the decision parameter as -
  + p0 = 2dx − dy
  + **Step 4.2.2** : At each yk along the line, starting at k = 0, perform the following test −
  + If pk < 0, the next point to plot is (xk,yk + 1) and pk+1 = pk + 2dx

else

* + plot (xk + 1,yk+1)
  + pk+1 = pk + 2dx − 2dy
  + **Step 4.2.3** : Repeat until y2>=y1.

**Step 5** : Exit.

**PROGRAM:**

#include <stdio.h>

#include <graphics.h>

#include <math.h>

// x -> Starting Point, y -> Ending Point, v -> Current Point

bool condition(int x, int y, int v)

{

if (x < y)

return v <= y;

else

return v >= y;

}

void drawLine(int a, int b, int c, int d)

{

int x = a, y = b;

int dx = (c - a), dy = (d - b);

float m = dy / (float)dx;

int p;

if (fabs(m) < 1)

{

p = 2 \* fabs(dy) - fabs(dx);

while (condition(a, c, x))

{

putpixel(x, y, WHITE);

printf("(%d, %d)\n", x, y);

if (x > c)

x--;

else if (x < c)

x++;

else

break;

if (p < 0)

{

p += 2 \* fabs(dy);

}

else

{

if (y < d)

y++;

else if (y > d)

y--;

p += 2 \* fabs(dy) - 2 \* fabs(dx);

}

}

}

else

{

p = 2 \* fabs(dx) - fabs(dy);

while (condition(b, d, y))

{

putpixel(x, y, WHITE);

printf("(%d, %d)\n", x, y);

if (y > d)

y--;

else if (y < d)

y++;

else

break;

if (p < 0)

{

p += 2 \* fabs(dx);

}

else

{

if (x < c)

x++;

else if (x > c)

x--;

p += 2 \* fabs(dx) - 2 \* fabs(dy);

}

}

}

}

int main()

{

int graphicsDriver = DETECT, graphicsMode;

int a, b, c, d;

printf("Enter starting points: ");

scanf("%d %d", &a, &b);

printf("Enter ending points: ");

scanf("%d %d", &c, &d);

printf("\n");

initgraph(&graphicsDriver, &graphicsMode, (char \*)"");

drawLine(a, b, c, d);

printf("\n");

getch();

closegraph();

return 0x1337;

}

**OUTPUT:**

