**Aim:** To implement Bresenham’s algorithms for drawing a line segment between two given end points.

**Objective:**

Draw a line using Bresenham's line algorithm that determines the points of an n-dimensional raster that should be selected to form a close approximation to a straight line between two points

**Theory:**

In Bresenham’s line algorithm pixel positions along the line path are obtained by determining the pixels i.e. nearer the line path at each step.

**Algorithm -**

Step 1: Except the two end points of Line from User.

Step 2: Calculate the slope(m) of the required Line.

Step 3: Identify the value of slope(m).

If slope(m) is Less than 1 i.e: m < 1

Calculate the constants dx, dy, 2dy, and (2dy – 2dx) and get the first value for the decision parameter as -

p0 = 2dy − dx

Step 4: 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

Repeat step 4 (dx - 1) times.

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

Calculate the constants dx, dy, 2dy, and (2dy – 2dx) and get the first value for the decision parameter as -

p0 = 2dx − dy

step 5: 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

Repeat step 5 (dy - 1) times.

Exit.

**Program –**

#include<stdio.h>

#include<conio.h>

#include<graphics.h>

int main()

{

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

int gd=DETECT;

int gm;

printf("Enter the x1 and y1");

scanf("%d %d",&x1,&y1);

printf("Enter the x2 and y2");

scanf("%d %d",&x2,&y2);

initgraph(&gd,&gm,"..//bgi");

dx=x2-x1;

dy=y2-y1;

pk=2\*dy-dx;

x=x1;

y=y1;

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

{

putpixel(x,y,9);

while(pk>=0)

{

y=y+1;

pk=pk-2\*dx;

}

x=x+1;

pk=pk+2\*dx;

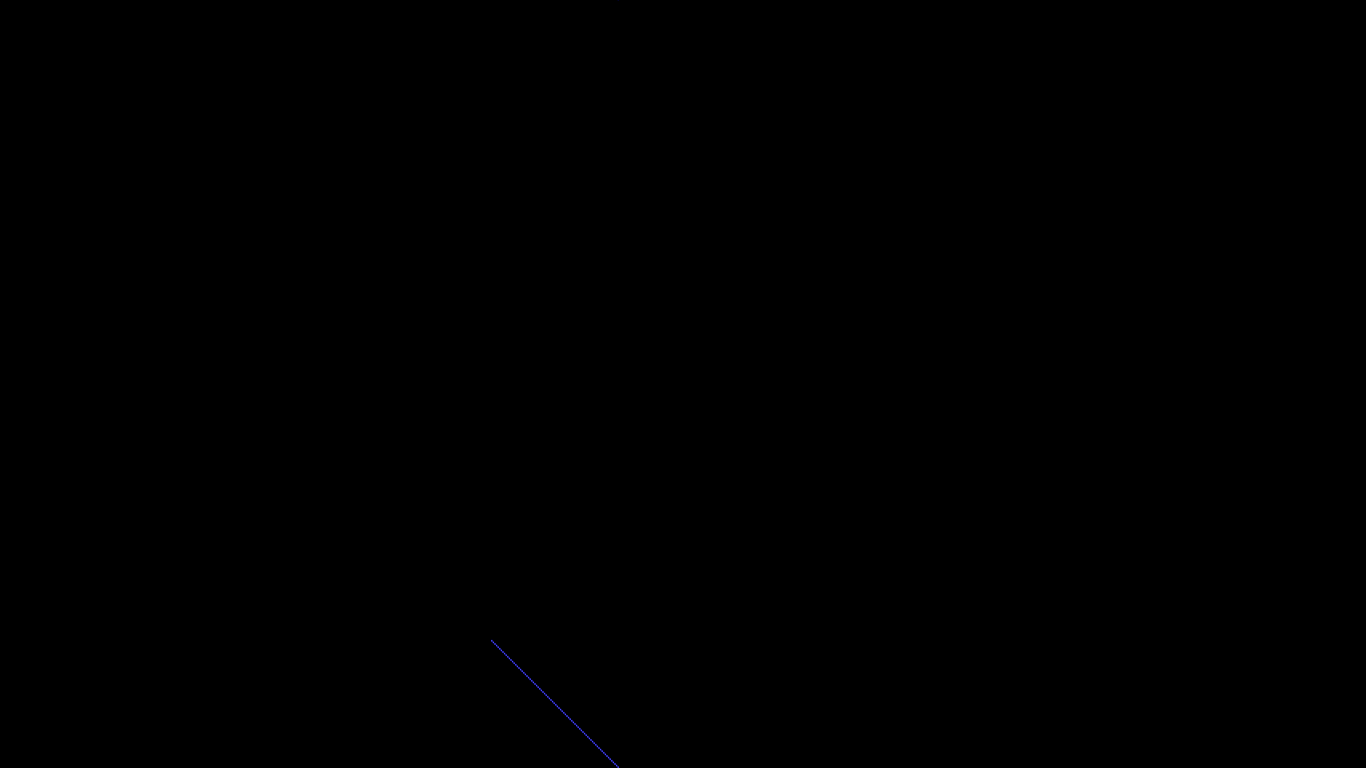
}

getch();

closegraph();

return 0;

}

**Output –** 

**Conclusion:** Comment on -

1. Pixel
2. Equation for line
3. Need of line drawing algorithm
4. Slow or fast