**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 – (x1,y1,x0,y0)**

**dx=x1-x0**

**dy=y1-y0**

**p0=2dy-dx**

**for k=0 to dx do**

**if pk<0 then**

**putpixel(xi+1,yi)**

**pn=pk+2dy**

**else**

**putpixel(xi+1,yi+1)**

**pn=pk+(2dy-2dx)**

**end**

**end**

**Program:**

#include<stdio.h>

#include<conio.h>

#include<graphics.h>

void main()

{

int x,y,dx,dy,p,x1,x2,y1,y2;

int gd=DETECT,gm;

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

printf("Enter the value ofx1,y1,x2,y2:" );

scanf("%d %d %d %d", &x1, &y1, &x2, &y2 );

x=x1;

y=y1;

dx=x2-x1;

dy=y2-y1;

p=2\*dy-dx;

while(x<=x2)

{

putpixel(x,y,CYAN);

x= x+1;

if(p<0)

{

p= p+ 2\*dy;

}

else

{

y=y+1;

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

}

}

getch();

closegraph();

}

**Output:**



**Conclusion:** Comment on -

1) Pixel:-Bresenham’s algorithm does not perform any rounding operation

2) Equation for line:-y=mx+c

3) Need of line drawing algorithm:-Involves cheaper operation like addition and subtraction

4) Slow or fast:-It is faster than DDA