Experiment No 1.

Aim: To implement DDA algorithms for drawing a line segment between two given end points.

Objective: Draw the line using (vector) generation algorithms which determine the pixels that should be turned ON are called as digital differential analyzer (DDA).It is one of the techniques for obtaining a rasterized straight line. This algorithm can be used to draw the line in all the quadrants.

Theory:
DDA algorithm is an incremental scan conversion method. Here we perform calculations at each step using the results from the preceding step. The characteristic of the DDA algorithm is to take unit steps along one coordinate and compute the corresponding values along the other coordinate. Digital Differential Analyzer (DDA) algorithm is the simple line generation algorithm which is explained step by step here.

Algorithm:

Step1: Start Algorithm

Step2: Declare x_1,y_1,x_2,y_2,dx,dy,x,y as integer variables.

Step3: Enter value of x_1,y_1,x_2,y_2.

Step4: Calculate dx = x_2-x_1

Step5: Calculate dy = y_2-y_1

Step6: If ABS (dx) > ABS (dy)
Then step = abs (dx)
Else

Step7: x_{inc}=dx/step
y_{inc}=dy/step
assign x = x_1
assign y = y_1

Step8: Set pixel (x, y)

Step9: x = x + x_{inc}
y = y + y_{inc}
Set pixels (Round (x), Round (y))

Step10: Repeat step 9 until $x = x_2$

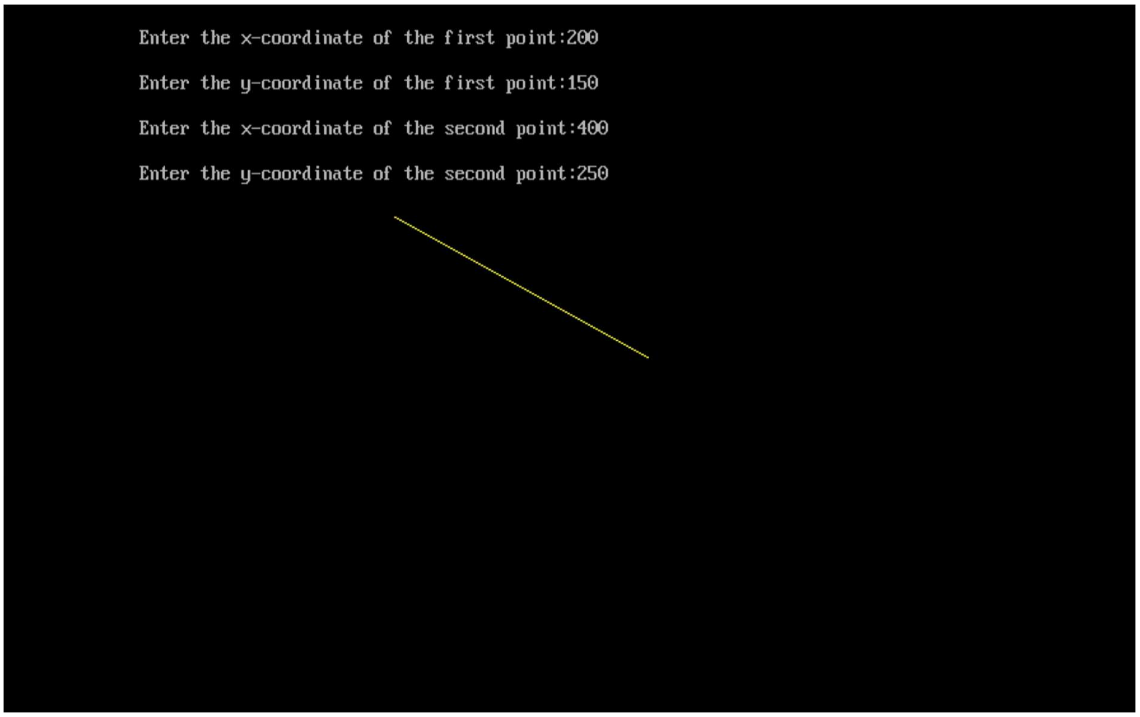
Step11: End Algorithm

Program:

```
#include<graphics.h>
#include<stdio.h>
#include<math.h>
#include<dos.h>
int main()
{
        float x,y,x1,y1,x2,y2,dx,dy,step;
        int i,gd=DETECT,gm;
        //detectgraph(&gd,&gm);
        initgraph(&gd,&gm,"");
        printf("\nEnter the x-coordinate of the first point:");
        scanf("%f",&x1);
        printf("\nEnter the y-coordinate of the first point:");
        scanf("%f",&y1);
        printf("\nEnter the x-coordinate of the second point:");
        scanf("%f",&x2);
        printf("\nEnter the y-coordinate of the second point:");
        scanf("%f",&y2);
        dx=abs(x2-x1);
        dy=abs(y2-y1);
        if(dx>dy)
        {
                step=dx;
        }
        else
        {
                step=dy;
        }
        dx=dx/step;
        dy=dy/step;
        x=x1;
        y=y1;
        i=1;
        while(i<=step)
        {
                putpixel(x,y,14);
                x=x+dx;
                y=y+dy;
                i=i+1;
                delay(100);
```

```
        }
        getch();
        closegraph();
}
```

Output:



Conclusion: Comment on -

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