Practical No:- 01.

Aim:→ To draw a line using the Bresenham's Line Drawing Algorithm in Turbo c graphics.

Theory:-

Bresenham's Line Drawing Algorithm is a simple and efficient algorithm used to draw a line between two given points (x_1, y_1) and (x_2, y_2) in a Computer graphic System. The algorithm calculates the point on the line path that are closest to the true line path and plots them to creat a line.

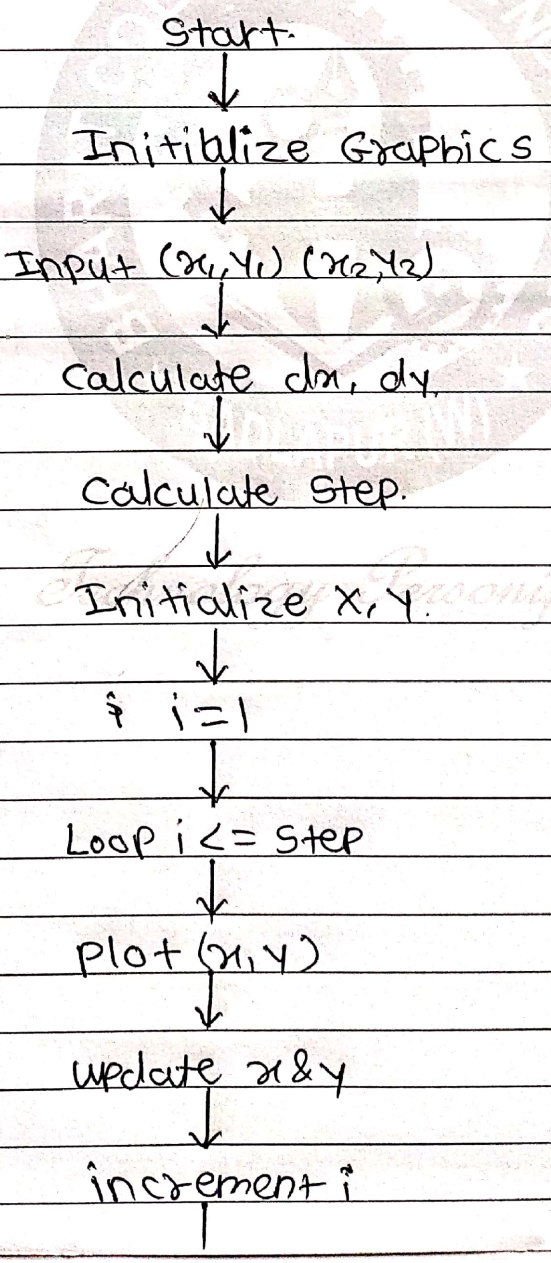
ALGORITHM:→

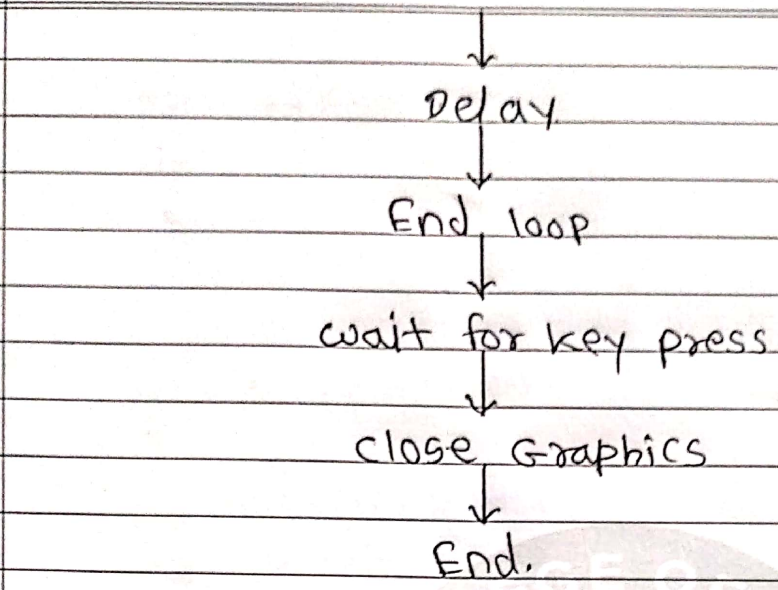
1. Start The program.
2. Initialize graphics mode using 'initgraph()'.
3. Prompt the user to enter the coordinates (x_1, y_1) and (x_2, y_2) of the line.
4. Calculate the absolute differences between x_2 & x_1 ('dx') and y_2 & y_1 ('dy')
5. Determine the number of steps ('step') required To draw the line which is the maximum of 'dx' and 'dy'.
6. Calculate the increment value for x ('dx') and y ('dy') based on 'step'.
7. Initialize 'x' and 'y' to the starting point ('x1' & y_1)
8. Initialize a counter variable 'i' to 1.
9. Enter a loop that runs for 'i' from 1 to 'step':
   a. plot the pixel at (x,y) using putpixel()
   b. update x & y by adding dx & dy respectively.

c. Increment i by 1.
d. add a delay to visualize the line drawing

10. End to loop.
11. wait for a key press using 'getch() to keep The graphics windoo open.
12. close the graphics mode using 'closegraph()'
13. End the program.

FLOWCHART:→





PROGRAM:-

```
# include <graphics.h>
# include <stdio.h>
# include <math.h>
# include <dos.h>
void main()
{
  float x, y, x1, x2, y1, y2, dx, dy, step;
  int i, gd=DETECT, gm;
  clrscr();
  initgraph(&gd, &gm, "c:\\turboc3\\bgi");
  printf("Enter the value of x1 & y1 : ");
  scanf("%f %f", &x1, &y1);
  printf("Enter the value of x2 & y2 : ");
  scanf("%f %f", &x2, &y2);
  dx = abs(x2-x1);
  dy = abs(y2-y1);
  if(dx>=dy)
  step=dx;
  else
  step=dy
```

```
dx = dx / step;
dy = dy / step;
x = xi
y = yi
i = 1;
While (i <= step)
{
   Putpixel (x, y, 5);
   x = x + dx
   y = y + dy;
   i = i + 1;
   delay (100);
}
getch ();
closegraph ();
}
```

PRACTICAL NO:- 02

AIM:- To draw a line using the Digital Differential Analyzer (DDA) Line Drawing algorithm in Turbo C graphics.

THEORY:→ The Digital differential Analyzer (DDA) Line Drawing Algorithm is a simple algorithm usedin to draw aline between to given points (x_1,y_1) & (x_2,y_2) in a computer graphics System. It calculates the slope of the line and incrementally plots pixels along the line path.

ALGORITHM:-

1. Start The program.
2. Initialize graphics mode using 'initgraph()'
3. prompt the user to enter the co-ordinates (x_1,y_1) and (x_2,y_2) of the line.
4. Calculate the absolute difference between x_2 & x_1 (dx) and y_2 & y_1 (dy)
5. Determine the starting point (x,y) based on wether x_1 is less than or greater than x_2
6. Calculate the initial decision parameter 'p' using the DDA algorithm.
7. Enter a loop that runs while 'x' is less than or equal to either x_1 or x_2:
   a. Check if 'p' is less than 0;
      - increment 'x' by 1.
      - keep 'y' unchanged.
      - Update 'p' as 'p+(2*dy)'.
   b. If 'p' is greater than or equal to 0;
      - increment both 'x' and 'y' by 1.
      - Update 'p' as 'p+(2*dy)-(2*dx)'.
   c. plot the pixel at (x,y) using 'putpixel()'.

8. End to loop
9. wait for a key press using 'getch()' to keep The graphic windoa open.
10. Close the graphics mode using 'closegraph().
11. End the program.

FLOWCHART:-

Start

Initialize Graphics

Input (x1, y1), (x2, y2)
&
Calculate dx, dy.

Determine (x, y).

Initialize p

Loop x<=x1 or x<=x2

Check if p<0

Increment x, keep y unchange

update p as p+(2*dy)

Else

Increment both x and y.

update p as p+(2*dy)-(2*dx).

~~plot~~

plot(x,y)

End loop

Wait for key press

close graphics

End.

PROGRAM:-

```
# include <stdio.h>
# include<conio.h>
# include <graphics.h>
# include <math.h>
void main()
{
  int dy, dx, x, y, x1, y1, x2, y2, p;
  int gd= DETECT, gm;
  detectgraph(&gd, &gm);
  initgraph(&gd, &gm, "C:\\Turboc3\\BGI");
  printf("Enter 1st Co-ordinate: ");
  Scanf("%d.%d", &x1, &y1);
  printf("Enter 2nd co-ordinate: ");
  Scanf("%d,%d", &x2, &y2);
  dx = abs(x2-x1)
  dy = abs(y2-y1)
  if(x1<x2.
```

```
{
    x=x1;
    y=y1;
}
if (x1>x2)
{
    x=x2
    y=y2
}
p=(2*dy)-dx;
While (x<=x1 || x<=x2)
{
    if (p<0)
    {
        x=x+1;
        y=y;
        p=p+(2*dy);
    }
    else
    {
        x=x+1;
        y=y+1;
        p=p+(2*dy)-(2*dx);
    }
    putpixel((int)x,(int)y,WHITE);
}
getch();
closegraph();
}
```