Assignment No:

Title/ Problem Statement:

Write C++/Java program for line drawing using DDA or Bresenhams algorithm with patterns such as solid, dotted, dashed and thick.

Objectives:

1. To understand Bresenham‘s or DDA Line drawing algorithms used for computer graphics.
2. To understand concept of different line styles using line algorithms

Theory :

DDA (Digital Differential Analyzer):

Scan conversion line algorithm based on Δx or Δy. Sample the line at unit interval in one co-ordinate and determine corresponding integer value. Faster method than y= mx+ b using but May specifies inaccurate pixel section. Rounding off operations and floating point arithmetic operations are time consuming. DDA is used in drawing straight line to form a line, triangle or polygon in computer graphics. DDA analyzes samples along the line at regular interval of one coordinate as the integer and for the other coordinate it rounds off the integer that is nearest to the line. Therefore as the line progresses it scan the first integer coordinate and round the second to nearest integer. Therefore a line drawn using DDA for x coordinate it will be x0 to x1 but for y coordinate it will be y=ax+ b and to draw function it will be fn(x, y rounded off).

i Plot	x	y	e
	5	5	0
1 (5, 5)	6	6	-8
2 (6, 6)	7	6	0
3 (7, 6)	8	7	-8
4 (8, 7)	9	7	0
5 (9, 7)	10	8	-8
6 (10, 8)	11	8	0
7 (11, 8)	12	9	-8
8 (12, 9)	13	9	

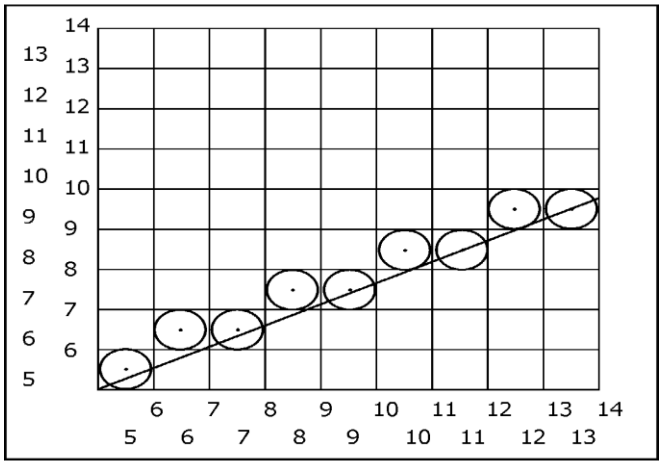

Fig DDA Line Algorithm

Advantages & disadvantages of DDA:-

• Faster than the direct use of the line equation and it does not do any floating point multiplication.

• Floating point Addition is still needed.

• Precision loss because of rounding off.

• Pixels drift farther apart if line is relatively larger.

```
#include<iostream>
#include<graphics.h>
using namespace std;
class linep
{
      public:
      int x1,y1,x2,y2;
      void drawl(int,int,int,int);
      void thick(int,int,int,int,int);
      void dotted(int,int,int,int);
      void dash(int ,int ,int,int);
};
void linep::thick(int x1,int y1,int x2,int y2,int w)
{
      int i,dx,dy,step,xinc,yinc;
      dx=x2-x1;
      dy=y2-y1;
      if(dy>=dx)
      {
            step=dy;
      }
      else
      {
            step=dx;
      }
      xinc=dx/step;
```

```
	yinc=dy/step;
	while(x1<=x2)
	{
		x1=x1+xinc+0.5;
		y1=y1+yinc+0.5;
		for(i=0;i<w;i++)
		{
			putpixel(x1+i,y1,BLUE);
		}
	}
}
void linep::drawl(int x1,int y1,int x2,int y2)
{
	int dx,dy,step,xinc,yinc;
	dx=x2-x1;
	dy=y2-y1;
	if(dy>=dx)
	{
		step=dy;
	}
	else
	{
		step=dx;
	}
	xinc=dx/step;
	yinc=dy/step;
	putpixel(x1,y1,BLUE);
	while(x1<=x2)
	{
		x1=x1+xinc+0.5;
		y1=y1+yinc+0.5;
		putpixel(x1,y1,BLUE);
	}
}
void linep:: dotted(int x1,int y1,int x2,int y2)
{
	int i=0,dx,dy,step,xinc,yinc;
	dx=x2-x1;
	dy=y2-y1;
	if(dy>=dx)
	{
		step=dy;
	}
	else
	{
		step=dx;
	}
	xinc=dx/step;
	yinc=dy/step;
	putpixel(x1,y1,BLUE);
	while(x1<=x2)
	{
```

```
            if(i%2==0)
            {
            putpixel(x1,y1,BLUE);
            }
            x1=x1+xinc+0.5;
            y1=y1+yinc+0.5;
            i++;
      }
}
void linep::dash(int x1,int y1,int x2,int y2)
{
      int i=0,dx,dy,step,xinc,yinc;
      dx=x2-x1;
      dy=y2-y1;
      if(dy>=dx)
      {
            step=dy;
      }
      else
      {
            step=dx;
      }
      xinc=dx/step;
      yinc=dy/step;
      putpixel(x1,y1,BLUE);
      while(x1<=x2)
      {
            if(i%5!=0)
            {
            putpixel(x1,y1,BLUE);
            }
            x1=x1+xinc+0.5;
            y1=y1+yinc+0.5;
            i++;
      }
}
int main()
{
      class linep p;
      int gd=DETECT ,gm,n,i,w;
      initgraph(&gd,&gm,NULL);
      cout<<"Enter co-ordinates of line x1,y1,x2,y2";
      cin>>p.x1>>p.y1>>p.x2>>p.y2;
      cout<<" \n1.thin line \n2.thick line \n3.dotted line \n4.dash
line \nenter your choice:";
      cin>>i;
      switch(i)
      {
      case 1:
            p.drawl(p.x1,p.y1,p.x2,p.y2);
            break;
      case 2:
```

```
                cout<<"enter width of line in pixel";
                cin>>w;
                p.thick(p.x1,p.y1,p.x2,p.y2,w);
                break;
           case 3:
                p.dotted(p.x1,p.y1,p.x2,p.y2);
                break;
           case 4:p.dash(p.x1,p.y1,p.x2,p.y2);
                break;
                default:
           cout<<"Enter proper choice";
           }
           getch();
           return 0;
}
```

Input:

Output:

Line Displayed in Different styles like thin, thick, dotted, dash.

Conclusion: Thus we have Studied inscribed and Circumscribed circles in the triangle.