EX 2

205001085
SABARIVASAN V

AIM:

To plot points that make up the line with endpoints (x0,y0) and (xn,yn) using the DDA line drawing algorithm. Case 1: +ve slope Left to Right line Case 2: +ve slope Right to Left line
Case 3: -ve slope Left to Right line

Case 4: -ve slope Right to Left line Each case has two subdivisions

(i) |m|<= 1 (ii) |m|>1
Note that all four cases of line drawing must be given as test cases.

Note that all four cases of line drawing must be given as test cases.

ALGORITHM:

- Inputlineendpoints,(x1,y1)and(x2,y2) setpixelatposition(x1,y1)
  calculateslopem=(y2-y1)/(x2-x1)
  For +ve slope (left to right)
- Case |m|≤1: Sample at unit x intervals and compute each successive y. Repeat the following steps until (x2, y2) is reached:
- **yk+1 = yk + m** where(m=y/ x)
  xk+1 =xk +1
  set pixel at position **(xk+1,Round(yk+1))**
- **Case |m|>1**: Sample at unit y intervals and compute each

successive x.

- ○ Repeat the following steps until (x2, y2) is reached: **xk+1 = xk+ 1/m**
- ○ **yk+1 =yk +1**
  set pixel at position **(Round(xk+1), yk+1)**

For +ve slope(right end point to left end point)
Case |m|≤1: Sample at unit x intervals and compute each successive y.
Repeat the following steps until (x2, y2) is reached:

yk+1 = yk - m where(m=y/ x)
xk+1 =xk -1
set pixel at position (xk-1,Round(yk-1))

Case |m|>1: Sample at unit y intervals and compute each successive x.
Repeat the following steps until (x2, y2) is reached:

xk+1 = xk -1/m
yk+1 =yk -1
set pixel at position (Round(xk-1), yk-1)

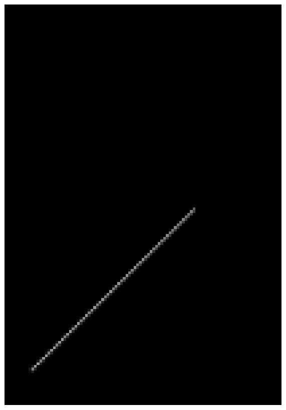
CODE :

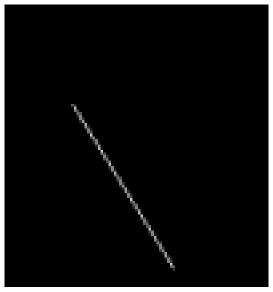
```
#include<GLUT/glut.h>
#include<iostream>
#include<cmath>
using namespace std;
void myInit() {
glClearColor(1.0,1.0,1.0,0.0);
glColor3f(0.0f,0.0f,0.0f);
glPointSize(10);
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
gluOrtho2D(-150.0,640.0,-150.0,480.0);
}
void myDisplay() {
glClear(GL_COLOR_BUFFER_BIT);
float x0,y0,xn,yn,x,y,m;
cin>>x0>>y0>>xn>>yn;
m = (yn-y0)/(xn-x0);
x=x0;
y=y0;
string p=to_string((int)x0);
int i=0,j=0;
while(i<p.length()){
glRasterPos2f(x0+j, y0-20);
glutBitmapCharacter(GLUT_BITMAP_TIMES_ROMAN_10, p[i]); i++;
j+=5;
}
glRasterPos2f(x0+j, y0-20);
glutBitmapCharacter(GLUT_BITMAP_TIMES_ROMAN_10, ',');
p=to_string((int)y0);
i=0;
while(i<p.length()){
glRasterPos2f(x0+j, y0-20);
```

```
glutBitmapCharacter(GLUT_BITMAP_TIMES_ROMAN_10, p[i]); i++;
j+=5;
}
p=to_string((int)xn);
i=0,j=0;
while(i<p.length()){
glRasterPos2f(xn+j, yn-20);
glutBitmapCharacter(GLUT_BITMAP_TIMES_ROMAN_10, p[i]); i++;
j+=5;
}
glRasterPos2f(xn+j, yn-20);
glutBitmapCharacter(GLUT_BITMAP_TIMES_ROMAN_10, ',');
p=to_string((int)yn);
i=0;
while(i<p.length()){
glRasterPos2f(xn+j, yn-20);
glutBitmapCharacter(GLUT_BITMAP_TIMES_ROMAN_10, p[i]); i++;
j+=5;
}
glBegin(GL_POINTS);
if(x0<xn){
if(m>0){
if(abs(m)<1){
while(x!=xn){
glVertex2d((int)round(x), (int)round(y));
x+=1;
y+=abs(m);
}
}
else{
while(y!=yn){
glVertex2d((int)round(x), (int)round(y));
x+=1/abs(m);
y+=1;
}
}
}
else{
if(abs(m)<1){
while(x!=xn){
glVertex2d((int)round(x), (int)round(y)); x+=1;
y-=abs(m);
}
}
else{
while(y!=yn){
glVertex2d((int)round(x), (int)round(y)); x+=1/abs(m);
y-=1;
}
}
}
}
else{
if(m>0){
if(abs(m)<1){
while(x!=xn){
glVertex2d((int)round(x), (int)round(y)); x-=1;
y-=abs(m);
}
}
else{
while(y!=yn){
glVertex2d((int)round(x), (int)round(y)); x-=1/abs(m);
y-=1;
}
}
}
else{
```

```
if(abs(m)<1){
while(x!=xn){
glVertex2d((int)round(x), (int)round(y)); x-=1;
y+=abs(m);
}
}
else{
while(y!=yn){
glVertex2d((int)round(x), (int)round(y)); x-=1/abs(m);
y+=1;
}
}
}
}
glEnd();
glFlush();
}
int main(int argc,char* argv[]) {
glutInit(&argc,argv);
glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
glutInitWindowSize(640,480);
glutInitWindowPosition((glutGet(GLUT_SCREEN_WIDTH)-640)/2,(glutGet(
GLUT_SCREEN_HEIGHT)-480)/2); glutCreateWindow("Second Exercise");
glutDisplayFunc(myDisplay);
myInit();
glutMainLoop();
return 1;
}
```

SAMPLE I/O:





LEARNING OUTCOME:
I learnt how to use bresenham’s line drawing algorithm in c++ usingopenGL library to draw a line.