**Task 1 : Write a c++ opengl program to draw a line using DDA line drawing algorithm. Use glBegin(GL\_POINTS);**

#include <windows.h>

#include <stdio.h>

#include <math.h>

#include <GL/glut.h>

double X1, Y1, X2, Y2;

float round\_value(float v)

{

return floor(v + 0.5);

}

void display(void) // display

{

double dx=(X2-X1);

double dy=(Y2-Y1);

double steps;

float xInc,yInc,x=X1,y=Y1;

/\* Find out whether to increment x or y \*/

steps=(abs(dx)>abs(dy))?(abs(dx)):(abs(dy));

xInc=dx/(float)steps;

yInc=dy/(float)steps;

/\* Clears buffers to preset values \*/

glClear(GL\_COLOR\_BUFFER\_BIT);

glPointSize(2.0);

/\* Plot the points \*/

glBegin(GL\_POINTS);

/\* Plot the first point \*/

glVertex2d(x,y);

int k;

/\* For every step, find an intermediate vertex \*/

for(k=0;k<steps;k++)

{

x+=xInc;

y+=yInc;

/\* printf("%0.6lf %0.6lf\n",floor(x), floor(y)); \*/

glVertex2d(round\_value(x), round\_value(y));

}

glEnd();

glFlush();

}

void Init()

{

/\* Set clear color to white \*/

glClearColor(1.0,1.0,1.0,0);

/\* Set fill color to black \*/

glColor3f(0.0,0.0,1.0);

/\* glViewport(0 , 0 , 640 , 480); \*/

/\* glMatrixMode(GL\_PROJECTION); \*/

/\* glLoadIdentity(); \*/

gluOrtho2D(0 , 50 , 0 , 50);

}

int main(int argc, char \*\*argv)

{

printf("Enter two end points of the line to be drawn:\n");

printf("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

printf("\nEnter Point1( X1 , Y1):\n");

scanf("%lf%lf",&X1,&Y1);

printf("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

printf("\nEnter Point2( X2 , Y2):\n");

scanf("%lf%lf",&X2,&Y2);

/\* Initialise GLUT library \*/

glutInit(&argc,argv);

/\* Set the initial display mode \*/

glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);

/\* Set the initial window position and size \*/

glutInitWindowPosition(0,0);

glutInitWindowSize(640,480);

/\* Create the window with title "DDA\_Line" \*/

glutCreateWindow("DDA\_Line");

/\* Initialize drawing colors \*/

Init();

/\* Call the displaying function \*/

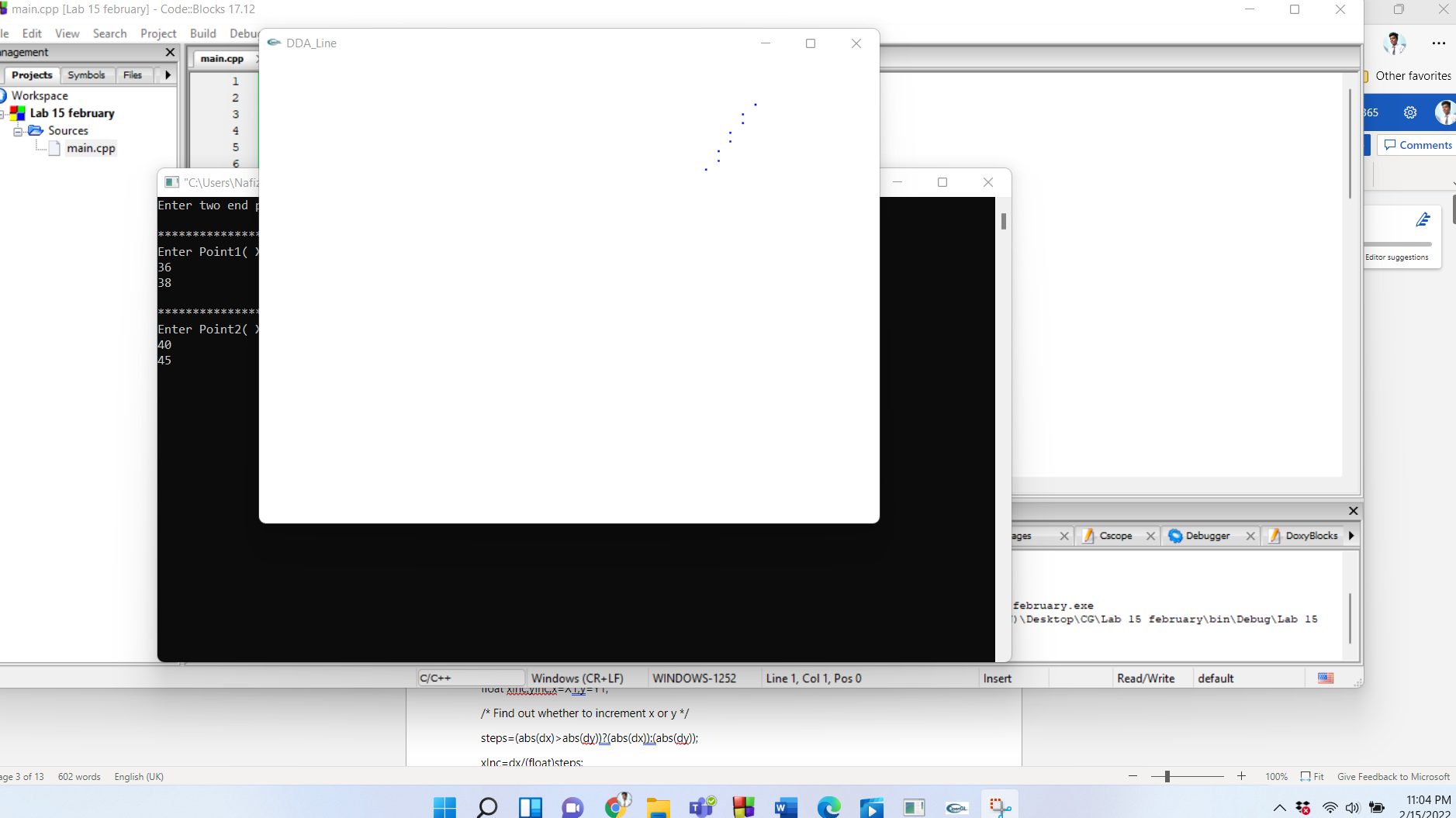
glutDisplayFunc(display);

/\* Keep displaying untill the program is closed \*/

glutMainLoop();

}

**OUTPUT**



**Task 2 : Write a c++ opengl program to draw a line using DDA line drawing algorithm. Use glBegin(GL\_LINES);**

#include <windows.h>

#include <stdio.h>

#include <math.h>

#include <GL/glut.h>

double X1, Y1, X2, Y2;

float round\_value(float v)

{

return floor(v + 0.5);

}

void display(void) // display

{

double dx=(X2-X1);

double dy=(Y2-Y1);

double steps;

float xInc,yInc,x=X1,y=Y1;

/\* Find out whether to increment x or y \*/

steps=(abs(dx)>abs(dy))?(abs(dx)):(abs(dy));

xInc=dx/(float)steps;

yInc=dy/(float)steps;

/\* Clears buffers to preset values \*/

glClear(GL\_COLOR\_BUFFER\_BIT);

glPointSize(2.0);

/\* Plot the points \*/

glBegin(GL\_LINE\_STRIP);

/\* Plot the first point \*/

glVertex2d(x,y);

int k;

/\* For every step, find an intermediate vertex \*/

for(k=0;k<steps;k++)

{

x+=xInc;

y+=yInc;

/\* printf("%0.6lf %0.6lf\n",floor(x), floor(y)); \*/

glVertex2d(round\_value(x), round\_value(y));

}

glEnd();

glFlush();

}

void Init()

{

/\* Set clear color to white \*/

glClearColor(1.0,1.0,1.0,0);

/\* Set fill color to black \*/

glColor3f(0.0,0.0,1.0);

/\* glViewport(0 , 0 , 640 , 480); \*/

/\* glMatrixMode(GL\_PROJECTION); \*/

/\* glLoadIdentity(); \*/

gluOrtho2D(0 , 50 , 0 , 50);

}

int main(int argc, char \*\*argv)

{

printf("Enter two end points of the line to be drawn:\n");

printf("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

printf("\nEnter Point1( X1 , Y1):\n");

scanf("%lf%lf",&X1,&Y1);

printf("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

printf("\nEnter Point2( X2 , Y2):\n");

scanf("%lf%lf",&X2,&Y2);

/\* Initialise GLUT library \*/

glutInit(&argc,argv);

/\* Set the initial display mode \*/

glutInitDisplayMode(GLUT\_SINGLE | GLUT\_RGB);

/\* Set the initial window position and size \*/

glutInitWindowPosition(0,0);

glutInitWindowSize(640,480);

/\* Create the window with title "DDA\_Line" \*/

glutCreateWindow("DDA\_Line");

/\* Initialize drawing colors \*/

Init();

/\* Call the displaying function \*/

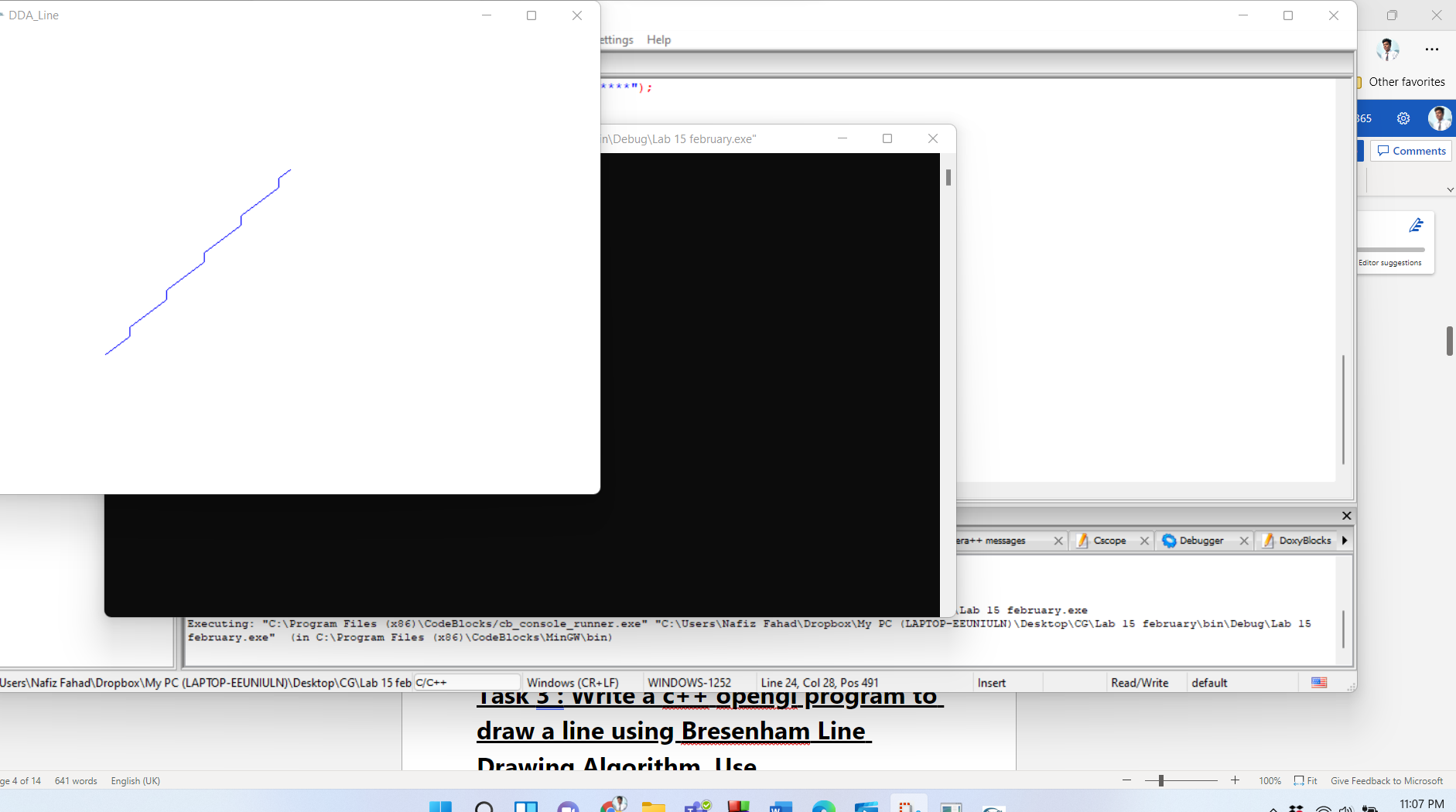
glutDisplayFunc(display);

/\* Keep displaying untill the program is closed \*/

glutMainLoop();

}

**OUTPUT**



**Task 3 : Write a c++ opengl program to draw a line using Bresenham Line Drawing Algorithm. Use glBegin(GL\_POINTS);**

#include<windows.h>

#include<GL/glu.h>

#include<GL/glut.h>

#include<math.h>

void Draw()

{

GLfloat x1=350,y1=30,x2=350,y2=400;

GLfloat M,p,dx,dy,x,y,t;

glClear(GL\_COLOR\_BUFFER\_BIT);

if((x2-x1)==0)

M = (y2-y1);

else

M = (y2-y1)/(x2-x1);

if(fabs(M)<1)

{

if(x1>x2)

{

t = x1;

x1 = x2;

x2 = t;

t = y1;

y1 = y2;

y2 = t;

}

dx = fabs(x2-x1);

dy = fabs(y2-y1);

p = 2\*dy-dx;

x=x1;

y=y1;

glBegin(GL\_POINTS);

while(x<=x2)

{

glVertex2f(x,y);

x=x+1;

if(p>=0)

{

if(M<1)

y=y+1;

else

y=y-1;

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

}

else

{

y=y;

p = p+2\*dy;

}

}

glEnd();

}

if(fabs(M)>=1)

{

if(y1>y2)

{

t = x1;

x1 = x2;

x2 = t;

t = y1;

y1 = y2;

y2 = t;

}

dx = fabs(x2-x1);

dy = fabs(y2-y1);

p = 2\*dx-dy;

x=x1;

y=y1;

glBegin(GL\_POINTS);

while(y<=y2)

{

glVertex2f(x,y);

y=y+1;

if(p>=0)

{

if(M>=1)

x=x+1;

else

x=x-1;

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

}

else

{

x=x;

p = p+2\*dx;

}

}

glEnd();

}

glFlush();

}

void MyInit()

{

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluOrtho2D(0,500,0,500);

glMatrixMode(GL\_MODELVIEW);

}

int main(int argC,char \*argV[])

{

glutInit(&argC,argV);

glutInitDisplayMode(GLUT\_RGB|GLUT\_SINGLE);

glutInitWindowPosition(0,0);

glutInitWindowSize(500,500);

glutCreateWindow("Brenham’s Line Drawing Algo");

MyInit();

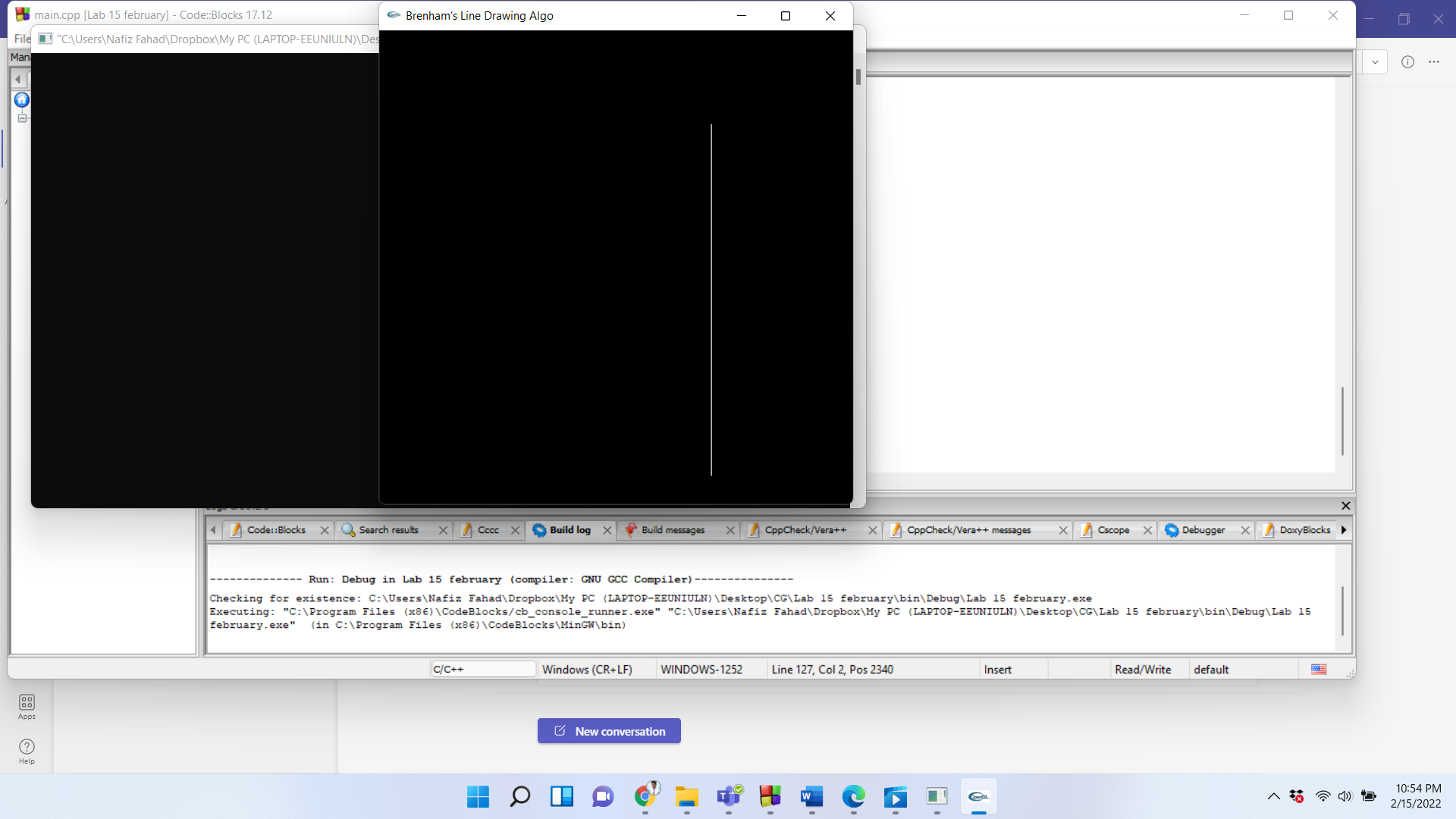
glutDisplayFunc(Draw);

glutMainLoop();

return 0;

}

**OUTPUT**



**Task 4 : Write a c++ opengl program to draw a line using Bresenham Line Drawing Algorithm. Use glBegin(GL\_LINES)**

#include<windows.h>

#include<GL/glu.h>

#include<GL/glut.h>

#include<math.h>

void Draw()

{

GLfloat x1=350,y1=340,x2=350,y2=400;

GLfloat M,p,dx,dy,x,y,t;

glClear(GL\_COLOR\_BUFFER\_BIT);

if((x2-x1)==0)

M = (y2-y1);

else

M = (y2-y1)/(x2-x1);

if(fabs(M)<1)

{

if(x1>x2)

{

t = x1;

x1 = x2;

x2 = t;

t = y1;

y1 = y2;

y2 = t;

}

dx = fabs(x2-x1);

dy = fabs(y2-y1);

p = 2\*dy-dx;

x=x1;

y=y1;

glBegin(GL\_LINE\_STRIP);

while(x<=x2)

{

glVertex2f(x,y);

x=x+1;

if(p>=0)

{

if(M<1)

y=y+1;

else

y=y-1;

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

}

else

{

y=y;

p = p+2\*dy;

}

}

glEnd();

}

if(fabs(M)>=1)

{

if(y1>y2)

{

t = x1;

x1 = x2;

x2 = t;

t = y1;

y1 = y2;

y2 = t;

}

dx = fabs(x2-x1);

dy = fabs(y2-y1);

p = 2\*dx-dy;

x=x1;

y=y1;

glBegin(GL\_LINE\_STRIP);

while(y<=y2)

{

glVertex2f(x,y);

y=y+1;

if(p>=0)

{

if(M>=1)

x=x+1;

else

x=x-1;

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

}

else

{

x=x;

p = p+2\*dx;

}

}

glEnd();

}

glFlush();

}

void MyInit()

{

glMatrixMode(GL\_PROJECTION);

glLoadIdentity();

gluOrtho2D(0,500,0,500);

glMatrixMode(GL\_MODELVIEW);

}

int main(int argC,char \*argV[])

{

glutInit(&argC,argV);

glutInitDisplayMode(GLUT\_RGB|GLUT\_SINGLE);

glutInitWindowPosition(0,0);

glutInitWindowSize(500,500);

glutCreateWindow("Brenham’s Line Drawing Algo");

MyInit();

glutDisplayFunc(Draw);

glutMainLoop();

return 0;

}

**OUTPUT**

