```
#include<windows.h>
#include<GL/glut.h>
#include<math.h>
#include<iostream>
#include <stdlib.h>
using namespace std;
class line
{
private:
    double ax1, ax2, ay1, ay2;
    double steps, dx, dy, x, y, xinc, yinc;
public:
    void get_data();
    void DDA();
    void simple_bresenham();
    void modified_bresenham();

} l;


void line::get_data()
{
    cout<<"Enter coordinates for line"<<endl;
    cout << "Enter x1 and y1 : ";
    cin>>ax1>>ay1;

    cout<<"Enter x2 and y2 : ";
    cin>>ax2>>ay2;

}

void line::simple_bresenham()
{
    int p = 0;
    dx = ax2 - ax1;
    dy = ay2 - ay1;
    p = (2 * dx) - dy;

    if (abs(dx) > abs(dy))
        steps = abs(dx);
    else (steps = abs(dy));

    glPointSize(5.0);

    for (int i = 0; i <= steps; i++)
    {
        if (p >= 0)
        {
            x = x + 1;
            y = y + 1;
```

```
            p = p + (2 * dy) - (2 * dx);
        }

        if (p < 0)
        {
            x = x + 1;
            y = y;
            p = p + (2 * dy);
        }
        glColor3f(0.0, 1.0, 0.0);
        glBegin(GL_POINTS);
        glVertex2i(x, y);
        glEnd();
    }
    glFlush();

}

void plotLineLow(int x0, int y0, int x1, int y1)
{
    int D = 0, dx = 0, dy = 0, x = 0, y = 0;
    dx = x1 - x0;
    dy = y1 - y0;
    int yi = 1;
    if (dy < 0)
    {
        yi = -1;
        dy = -dy;
    }

    D = 2 * dy - dx;
    y = y0;

    glPointSize(5.0);

    for (x = x0; x < x1; x++)
    {
        glColor3f(0.0, 0.0, 1.0);
        glBegin(GL_POINTS);
        glVertex2i(x, y);
        glEnd();
        if (D > 0)
        {
            y = y + yi;
            D = D - 2 * dx;
        }
        D = D + 2 * dy;
    }

    glFlush();
```

```
}

void plotLineHigh(int x0, int y0, int x1, int y1)
{
    int D = 0, dx = 0, dy = 0, x = 0, y = 0;
    dx = x1 - x0;
    dy = y1 - y0;
    int xi = 1;
    if (dx < 0)
    {
        xi = -1;
        dx = -dx;
    }
    D = 2 * dx - dy;
    x = x0;

    glPointSize(5.0);


    for (y = y0; y < y1; y++)
    {
        glColor3f(1.0, 0.0, 1.0);
        glBegin(GL_POINTS);
        glVertex2i(x, y);
        glEnd();
        if (D > 0)
        {
            x = x + xi;
            D = D - 2 * dy;
        }
        D = D + 2 * dx;
    }

    glFlush();
}

void line::modified_bresenham()
{
    if (abs(ay2 - ay1) < abs(ax2 - ax1))
    {
        if (ax1 > ax2)
            plotLineLow(ax2, ay2, ax1, ay1);
        else
            plotLineLow(ax1, ay1, ax2, ay2);
    }

    else
    {
        if (y0 > y1)
            plotLineHigh(ax2, ay2, ax1, ay1);
```

```
        else
            plotLineHigh(ax1, ay1, ax2, ay2);
    }
}

void line::DDA()
{
    dx = ax2 - ax1;
    dy = ay2 - ay1;

    if (abs(dx) > abs(dy))
        steps = abs(dx);
    else (steps = abs(dy));

    xinc = dx / steps;
    yinc = dy / steps;
    x = ax1;
    y = ay1;

    glPointSize(3.0);

    glBegin(GL_POINTS);
    glVertex2i(x, y);
    glEnd();

    for (int i = 0; i <= steps; i++)
    {
        x = x + xinc;
        y = y + yinc;
        glColor3f(1.0, 0.0, 0.0);
        glBegin(GL_POINTS);
        glVertex2i(x, y);
        glEnd();
    }
    glFlush();
}

void lines(int item)
{
    if (item == 1)
        l.DDA();
    else if (item == 2)
        l.simple_bresenham();
    else if (item == 3)
        l.modified_bresenham();
}
void keyboard(unsigned char key, int x, int y)
{

    if (key == 'd')
```

```
        l.DDA();
    else if (key == 's')
        l.simple_bresenham();
    else if(key == 'm')
        l.modified_bresenham();
}

void display(void)
{


}


void init(void)
{
    glClearColor(1.0, 1.0, 1.0, 0.0);
    glClear(GL_COLOR_BUFFER_BIT);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluOrtho2D(-500, 500, -500, 500);
}


int main(int argc, char* argv[])
{
    cout<<"press 'd' for DDA line generation and 'b' for simple bresenham
generation"<<endl;
    l.get_data();
    glutInit(&argc, argv);
    glutInitWindowSize(500, 500);
    glutInitWindowPosition(5, 5);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    glutCreateWindow("Line Generator 7241_Priya");
    init();

    glutKeyboardFunc(keyboard);

    glutCreateMenu(lines);
    glutAddMenuEntry("DDA", 1);
    glutAddMenuEntry("Simple bresenham", 2);
    glutAddMenuEntry("Modified bresenham", 3);
    glutAttachMenu(GLUT_RIGHT_BUTTON);

    glutDisplayFunc(display);
    glutMainLoop();
    return 0;
}
```