#include <graphics.h>

#include <iostream>

using namespace std;

class Pt

{

protected:

int x1, y1, color;

public:

Pt()

{

x1 = 0;

y1 = 0;

color = WHITE;

}

void setco(int x0, int y0)

{

x1 = x0;

y1 = y0;

}

void draw()

{

putpixel(x1, y1, color);

}

};

class dline : public Pt

{

private:

int x2, y2;

public:

dline() : Pt()

{

x2 = 0;

y2 = 0;

}

void setline(int x0, int y0, int xx, int yy)

{

Pt::setco(x0, y0);

x2 = xx;

y2 = yy;

}

void drawl()

{

int x, y, dx, dy, steps, i = 0;

float xinc, yinc;

dx = x2 - x1;

dy = y2 - y1;

if (abs(dx) > abs(dy))

steps = abs(dx);

else

steps = abs(dy);

xinc = dx / (float)steps;

yinc = dy / (float)steps;

x = x1;

y = y1;

Pt::setco(x, y);

Pt::draw();

for (i = 1; i <= steps; i++)

{

x = x + xinc;

y = y + yinc;

Pt::setco((int)x, (int)y); // Ensure coordinates are integers

Pt::draw();

}

}

void cal(int x1, int y1, int x3, int y3)

{

setline(x1, y1, x3, y1);

drawl();

setline(x3, y1, x3, y3);

drawl();

setline(x3, y3, x1, y3);

drawl();

setline(x1, y3, x1, y1);

drawl();

int xmid1, ymid1, xmid2, ymid2, xmid3, ymid3, xmid4, ymid4;

xmid1 = (x1 + x3) / 2;

ymid1 = y1;

xmid2 = x3;

ymid2 = (y1 + y3) / 2;

setline(xmid1, ymid1, xmid2, ymid2);

drawl();

xmid3 = (x1 + x3) / 2;

ymid3 = y3;

setline(xmid2, ymid2, xmid3, ymid3);

drawl();

xmid4 = x1;

ymid4 = (y1 + y3) / 2;

setline(xmid3, ymid3, xmid4, ymid4);

drawl();

setline(xmid4, ymid4, xmid1, ymid1);

drawl();

int xf1, yf1, xf2, yf2, xf3, yf3, xf4, yf4;

xf1 = (xmid1 + xmid4) / 2;

yf1 = (ymid1 + ymid4) / 2;

xf2 = (xmid1 + xmid2) / 2;

yf2 = (ymid1 + ymid2) / 2;

setline(xf1, yf1, xf2, yf2);

drawl();

xf3 = (xmid2 + xmid3) / 2;

yf3 = (ymid2 + ymid3) / 2;

setline(xf2, yf2, xf3, yf3);

drawl();

xf4 = (xmid3 + xmid4) / 2;

yf4 = (ymid3 + ymid4) / 2;

setline(xf3, yf3, xf4, yf4);

drawl();

setline(xf4, yf4, xf1, yf1);

drawl();

}

};

int main()

{

int x1, y1, x3, y3;

int gd = DETECT, gm;

// Initialize graphics mode

initgraph(&gd, &gm, NULL);

dline dda;

cout << "\nEnter the coordinates of the endpoints below\n";

cout << "Enter x1: ";

cin >> x1;

cout << "Enter y1: ";

cin >> y1;

cout << "Enter x3: ";

cin >> x3;

cout << "Enter y3: ";

cin >> y3;

// Call to draw the pattern

dda.cal(x1, y1, x3, y3);

// Wait and close the graphics window

delay(50000);

closegraph();

return 0;

}