/\*B21 rhombus : Write a C++ program to implement translation, sheer, rotation and scaling transformations on equilateral rhombus\*/

#include <iostream>

#include <graphics.h>

#include <stdlib.h>

#include <math.h>

using namespace std;

int main()

{

int x1, x2, x3, y1, y2, y3, x4, y4, nx1, nx2, nx3, ny1, ny2, ny3, nx4, ny4, shx, shy, poly, ch\_r, ch\_e;

float sx, sy, xt, yt, r;

float t;

char ch;

int gdriver = DETECT, gmode;

do

{

cout << "\n1.Rhombus or 2.equilateral triangle";

cin >> poly;

if (poly == 1)

{

x1 = 15;

y1 = 15;

x2 = 75;

y2 = 45;

x3 = 105;

y3 = 105;

x4 = 45;

y4 = 75;

cout << "\n1.Translation\n2.rotaion \n3.scaling\n4.shear\n5.exit\nenter your choice";

cin >> ch\_r;

switch (ch\_r)

{

case 1:

cout << "\nenter the translation factor x";

cin >> xt;

cout << "\nenter the translation factor y";

cin >> yt;

nx1 = x1 + xt;

ny1 = y1 + yt;

nx2 = x2 + xt;

ny2 = y2 + yt;

nx3 = x3 + xt;

ny3 = y3 + yt;

nx4 = x4 + xt;

ny4 = y4 + yt;

initgraph(&gdriver, &gmode, NULL);

setcolor(1);

line(x1, y1, x2, y2);

line(x2, y2, x3, y3);

line(x3, y3, x4, y4);

line(x4, y4, x1, y1);

setcolor(WHITE);

line(nx1, ny1, nx2, ny2);

line(nx2, ny2, nx3, ny3);

line(nx3, ny3, nx4, ny4);

line(nx4, ny4, nx1, ny1);

delay(50000);

closegraph();

break;

case 2:

cout << "\nenter the angle of rotation";

cin >> r;

t = 3.14 \* r / 180;

nx1 = abs(x1 \* cos(t) - y1 \* sin(t));

ny1 = abs(x1 \* sin(t) + y1 \* cos(t));

nx2 = abs(x2 \* cos(t) - y2 \* sin(t));

ny2 = abs(x2 \* sin(t) + y2 \* cos(t));

nx3 = abs(x3 \* cos(t) - y3 \* sin(t));

ny3 = abs(x3 \* sin(t) + y3 \* cos(t));

nx4 = abs(x4 \* cos(t) - y4 \* sin(t));

ny4 = abs(x4 \* sin(t) + y4 \* cos(t));

initgraph(&gdriver, &gmode, NULL);

setcolor(1);

line(x1, y1, x2, y2);

line(x2, y2, x3, y3);

line(x3, y3, x4, y4);

line(x4, y4, x1, y1);

setcolor(WHITE);

line(nx1, ny1, nx2, ny2);

line(nx2, ny2, nx3, ny3);

line(nx3, ny3, nx4, ny4);

line(nx4, ny4, nx1, ny1);

delay(500);

closegraph();

break;

case 3:

cout << "\nenter the scaling factor x";

cin >> sx;

cout << "\nenter the scaling factor y";

cin >> sy;

nx1 = x1 \* sx;

ny1 = y1 \* sy;

nx2 = x2 \* sx;

ny2 = y2 \* sy;

nx3 = x3 \* sx;

ny3 = y3 \* sy;

nx4 = x4 \* sx;

ny4 = y4 \* sy;

initgraph(&gdriver, &gmode, NULL);

setcolor(1);

line(x1, y1, x2, y2);

line(x2, y2, x3, y3);

line(x3, y3, x4, y4);

line(x4, y4, x1, y1);

setcolor(WHITE);

line(nx1, ny1, nx2, ny2);

line(nx2, ny2, nx3, ny3);

line(nx3, ny3, nx4, ny4);

line(nx4, ny4, nx1, ny1);

delay(500);

closegraph();

break;

case 4:

cout << "\nenter the shear factor x";

cin >> shx;

cout << "\nenter the shear factor y";

cin >> shy;

nx1 = abs(x1 + shx \* y1);

ny1 = abs(y1 + shy \* x1);

nx2 = abs(x2 + shx \* y2);

ny2 = abs(y2 + shy \* x2);

nx3 = abs(x3 + shx \* y3);

ny3 = abs(y3 + shy \* x3);

nx4 = abs(x4 + shx \* y4);

ny4 = abs(y4 + shx \* x4);

initgraph(&gdriver, &gmode, NULL);

setcolor(1);

line(x1, y1, x2, y2);

line(x2, y2, x3, y3);

line(x3, y3, x4, y4);

line(x4, y4, x1, y1);

setcolor(WHITE);

line(nx1, ny1, nx2, ny2);

line(nx2, ny2, nx3, ny3);

line(nx3, ny3, nx4, ny4);

line(nx4, ny4, nx1, ny1);

delay(500);

closegraph();

break;

case 5:

break;

default:

cout << "enter the correct choice";

}

}

if (poly == 2)

{

x1 = 100;

y1 = 200;

x2 = 200;

y2 = 200;

x3 = 150;

y3 = 113;

cout << "\n1.Translation\n2.rotaion \n3.scaling\n4.shear\n5.exit\nenter your choice";

cin >> ch\_e;

switch (ch\_e)

{

case 1:

cout << "\nenter the translation factor x";

cin >> xt;

cout << "\nenter the translation factor y";

cin >> yt;

nx1 = x1 + xt;

ny1 = y1 + yt;

nx2 = x2 + xt;

ny2 = y2 + yt;

nx3 = x3 + xt;

ny3 = y3 + yt;

initgraph(&gdriver, &gmode, NULL);

setcolor(1);

line(x1, y1, x2, y2);

line(x2, y2, x3, y3);

line(x3, y3, x1, y1);

setcolor(WHITE);

line(nx1, ny1, nx2, ny2);

line(nx2, ny2, nx3, ny3);

line(nx3, ny3, nx1, ny1);

delay(500);

closegraph();

break;

case 2:

cout << "\nenter the angle of rotation";

cin >> r;

t = 3.14 \* r / 180;

nx1 = abs(x1 \* cos(t) - y1 \* sin(t));

ny1 = abs(x1 \* sin(t) + y1 \* cos(t));

nx2 = abs(x2 \* cos(t) - y2 \* sin(t));

ny2 = abs(x2 \* sin(t) + y2 \* cos(t));

nx3 = abs(x3 \* cos(t) - y3 \* sin(t));

ny3 = abs(x3 \* sin(t) + y3 \* cos(t));

initgraph(&gdriver, &gmode, NULL);

setcolor(1);

line(x1, y1, x2, y2);

line(x2, y2, x3, y3);

line(x3, y3, x1, y1);

setcolor(WHITE);

line(nx1, ny1, nx2, ny2);

line(nx2, ny2, nx3, ny3);

line(nx3, ny3, nx1, ny1);

delay(500);

closegraph();

break;

case 3:

cout << "\nenter the scaling factor x";

cin >> sx;

cout << "\nenter the scaling factor y";

cin >> sy;

nx1 = x1 \* sx;

ny1 = y1 \* sy;

nx2 = x2 \* sx;

ny2 = y2 \* sy;

nx3 = x3 \* sx;

ny3 = y3 \* sy;

initgraph(&gdriver, &gmode, NULL);

setcolor(1);

line(x1, y1, x2, y2);

line(x2, y2, x3, y3);

line(x3, y3, x1, y1);

setcolor(WHITE);

line(nx1, ny1, nx2, ny2);

line(nx2, ny2, nx3, ny3);

line(nx3, ny3, nx1, ny1);

delay(500);

closegraph();

break;

case 4:

cout << "\nenter the shear factor x";

cin >> shx;

cout << "\nenter the shear factor y";

cin >> shy;

nx1 = abs(x1 + shx \* y1);

ny1 = abs(y1 + shy \* x1);

nx2 = abs(x2 + shx \* y2);

ny2 = abs(y2 + shy \* x2);

nx3 = abs(x3 + shx \* y3);

ny3 = abs(y3 + shy \* x3);

initgraph(&gdriver, &gmode, NULL);

setcolor(1);

line(x1, y1, x2, y2);

line(x2, y2, x3, y3);

line(x3, y3, x1, y1);

setcolor(WHITE);

line(nx1, ny1, nx2, ny2);

line(nx2, ny2, nx3, ny3);

line(nx3, ny3, nx1, ny1);

delay(500);

closegraph();

break;

case 5:

break;

default:

cout << "enter the correct choice";

}

}

cout << "Would you like to continue(y/n)";

cin >> ch;

} while (ch == 'y');

delay(500);

closegraph();

}