#include<iostream>

#include<stdlib.h>

#include<graphics.h>

#include<math.h>

using namespace std;

class POLYGON

{ private:

int p[10][10],Trans\_result[10][10],Trans\_matrix[10][10]; float Rotation\_result[10][10],Rotation\_matrix[10][10]; float Scaling\_result[10][10],Scaling\_matrix[10][10]; float Shearing\_result[10][10],Shearing\_matrix[10][10]; int Reflection\_result[10][10],Reflection\_matrix[10][10];

public:

int accept\_poly(int [][10]); void draw\_poly(int [][10],int); void draw\_polyfloat(float [][10],int);

void matmult(int [][10],int [][10],int,int,int,int [][10]); void matmultfloat(float [][10],int [][10],int,int,int,float [][10]);

void shearing(int [][10],int); void scaling(int [][10],int); void rotation(int [][10],int); void translation(int [][10],int);

void reflection(int [][10],int);

};

int POLYGON :: accept\_poly(int p[][10])

{

int i,n;

cout<<"\n\n\t\tEnter no.of vertices:";

cin>>n;

for(i=0;i<n;i++)

{

cout<<"\n\n\t\tEnter (x,y)Co-ordinate of point P"<<i<<": ";

cin >> p[i][0] >> p[i][1]; p[i][2] = 1;

}

for(i=0;i<n;i++)

{

cout<<"\n";

for(int j=0;j<3;j++)

{

cout<<p[i][j]<<"\t";

}

}

return n;

}

void POLYGON :: draw\_poly(int p[][10], int n)

{

int i,gd = DETECT,gm; initgraph(&gd,&gm,NULL); line(320,0,320,480); line(0,240,640,240);

for(i=0;i<n;i++)

{

if(i<n-1)

{

line(p[i][0]+320, -p[i][1]+240, p[i+1][0]+320, -p[i+1][1]+240);

}

else

line(p[i][0]+320, -p[i][1]+240, p[0][0]+320, -p[0][1]+240);

}

delay(3000);

}

void POLYGON :: draw\_polyfloat(float p[][10], int n)

{

int i,gd = DETECT,gm; initgraph(&gd,&gm,NULL); line(320,0,320,480); line(0,240,640,240);

for(i=0;i<n;i++)

{

if(i<n-1)

{

line(p[i][0]+320, -p[i][1]+240, p[i+1][0]+320, -p[i+1][1]+240);

}

else

line(p[i][0]+320, -p[i][1]+240, p[0][0]+320, -p[0][1]+240);

}

//delay(8000);

}

void POLYGON :: translation(int p[10][10],int n)

{

int tx,ty,i,j; int i1,j1,k1,r1,c1,c2;

r1=n;c1=c2=3;

cout << "\n\n\t\tEnter X-Translation tx: "; cin >> tx;

cout << "\n\n\t\tEnter Y-Translation ty: "; cin >> ty; for(i=0;i<3;i++) for(j=0;j<3;j++) Trans\_matrix[i][j] = 0;

Trans\_matrix[0][0] = Trans\_matrix[1][1] = Trans\_matrix[2][2] = 1;

Trans\_matrix[2][0] = tx;

Trans\_matrix[2][1] = ty;

for(i1=0;i1<10;i1++) for(j1=0;j1<10;j1++) Trans\_result[i1][j1] = 0; for(i1=0;i1<r1;i1++) for(j1=0;j1<c2;j1++)

for(k1=0;k1<c1;k1++)

Trans\_result[i1][j1] = Trans\_result[i1][j1]+(p[i1][k1] \* Trans\_matrix[k1][j1]); cout << "\n\n\t\tPolygon after Translationâ€¦";

draw\_poly(Trans\_result,n);

}

void POLYGON :: rotation(int p[][10],int n)

{

float type,Ang,Sinang,Cosang;

int i,j; int i1,j1,k1,r1,c1,c2;

r1=n;c1=c2=3;

cout << "\n\n\t\tEnter the angle of rotation in degrees: ";

cin >> Ang;

cout << "\n\n \*\*\*\* Rotation Types \*\*\*\*";

cout << "\n\n\t\t1.Clockwise Rotation \n\n\t\t2.Anti-Clockwise Rotation "; cout << "\n\n\t\tEnter your choice(1-2): "; cin >> type;

Ang = (Ang \* 6.2832)/360;

Sinang = sin(Ang); Cosang = cos(Ang);

cout<<"Mark1";

for(i=0;i<3;i++) for(j=0;j<3;j++)

Rotation\_matrix[i][j] = 0;

cout<<"Mark2";

Rotation\_matrix[0][0] = Rotation\_matrix[1][1] = Cosang;

Rotation\_matrix[0][1] = Rotation\_matrix[1][0] = Sinang;

Rotation\_matrix[2][2] = 1; if(type == 1)

Rotation\_matrix[0][1] = -Sinang;

else

Rotation\_matrix[1][0] = -Sinang;

for(i1=0;i1<10;i1++)

for(j1=0;j1<10;j1++) Rotation\_result[i1][j1] = 0; for(i1=0;i1<r1;i1++) for(j1=0;j1<c2;j1++) for(k1=0;k1<c1;k1++)

Rotation\_result[i1][j1] = Rotation\_result[i1][j1]+(p[i1][k1] \*

Rotation\_matrix[k1][j1]);

cout << "\n\n\t\tPolygon after Rotationâ€¦";

for(i=0;i<n;i++)

{

cout<<"\n";

for(int j=0;j<3;j++)

{

cout<<Rotation\_result[i][j]<<"\t";

}

}

draw\_polyfloat(Rotation\_result,n);

}

void POLYGON :: scaling(int p[][10],int n)

{

float Sx,Sy;

int i,j; int i1,j1,k1,r1,c1,c2; r1=n;c1=c2=3;

cout<<"\n\n\t\tEnter X-Scaling Sx: ";

cin>>Sx;

cout<<"\n\n\t\tEnter Y-Scaling Sy: ";

cin>>Sy;

for(i=0;i<3;i++)

{

for(j=0;j<3;j++)

{

Scaling\_matrix[i][j] = 0;

}

}

Scaling\_matrix[0][0] = Sx;

Scaling\_matrix[0][1] = 0;

Scaling\_matrix[0][2] = 0;

Scaling\_matrix[1][0] = 0;

Scaling\_matrix[1][1] = Sy;

Scaling\_matrix[1][2] = 0;

Scaling\_matrix[2][0] = 0;

Scaling\_matrix[2][1] = 0;

Scaling\_matrix[2][2] = 1;

for(i1=0;i1<10;i1++)

for(j1=0;j1<10;j1++) Scaling\_result[i1][j1] = 0; for(i1=0;i1<r1;i1++) for(j1=0;j1<c2;j1++) for(k1=0;k1<c1;k1++)

Scaling\_result[i1][j1] = Scaling\_result[i1][j1]+(p[i1][k1] \*

Scaling\_matrix[k1][j1]);

cout<<"\n\n\t\tPolygon after Scalingâ€¦"; draw\_polyfloat(Scaling\_result,n);

}

int main()

{

int ch,n,p[10][10];

POLYGON p1;

cout<<"\n\n \*\*\*\* 2-D TRANSFORMATION \*\*\*\*";

n= p1.accept\_poly(p);

cout <<"\n\n\t\tOriginal Polygon â€¦";

p1.draw\_poly(p,n);

do {

int ch;

cout<<"\n\n \*\*\*\* 2-D TRANSFORMATION \*\*\*\*";

cout<<"\n\n\t\t1.Translation \n\n\t\t2.Scaling \n\n\t\t3.Rotation \n\n\t\t4.Exit"; cout<<"\n\n\tEnter your choice(1-6):";

cin>>ch;

switch(ch)

{

case 1:

//cout<<"case1"; p1.translation(p,n);

break;

case 2:

cout<<"case2"; p1.scaling(p,n);

break;

case 3:

cout<<"case3"; p1.rotation(p,n);

break;

case 4: exit(0);

}

}while(1);

return 0;

}