**ASSIGNMENT NO. :- 04**

**TITLE :- Write C++ program to draw 2-D object and perform following basic transformations, Scaling b) Translation c) Rotation. Apply the concept of operator overloading.**

#include<iostream>

#include<math.h>

#include<graphics.h>

using namespace std;

class matrix

{

public:

int n,i,j,tx,ty,k,sum,sx,sy;

double a[6][3],b[6][3],mult[6][3],mat3[6][3];

double p,q,r;

double ang=0,angle=0;

public:

void get()

{

cout<<"\n enter the number of vertices of polygon : ";

cin>>n;

// cout<<"n Entering user matix\n";

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

{

cout<<"enter x n y co ordinates";

cin>>b[i][0];

cin>>b[i][1];

b[i][2]=1;

}

//display object matrix

cout<<"\n original co ordinates are"<<"\n";

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

{

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

{

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

}cout<<"\n";

}

}

void identitymat()

{

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

{

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

{

if(i==j)

{

a[i][j]=1;

}

else

{

a[i][j]=0;

}

}

}

}

void trans()

{

cout<<"enter values of tx and ty";

cin>>tx>>ty;

a[2][0]=tx;

a[2][1]=ty;

cout<<"matrix is"<<"\n";

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

{

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

{

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

}cout<<"\n";

}

}

void scale()

{

cout<<"\n Enter the values of sx and sy";

cin>>sx>>sy;

a[0][0]=sx;

a[1][1]=sy;

cout<<"\n Matrix is:"<<"\n";

//To display scaling matrix

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

{

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

{

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

}cout<<"\n";

}

}

void rot()

{

cout<<"Enter the angle";

cin>>ang;

angle=(ang\*3.142)/180;

q=sin(angle);

p=cos(angle);

r=-sin(angle);

a[0][0]=p;

a[0][1]=q;

a[1][0]=r;

a[1][1]=p;

cout<<"tranformation matrix is"<<"\n";

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

{

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

{

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

}cout<<"\n";

}

}

void multi()

{

cout<<"\nMultiplying two matrices...";

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

{

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

{

sum=0;

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

{

sum = sum + b[i][k] \* a[k][j];

}

mat3[i][j] = sum;

}

}

}

void display()

{

cout<<"\nMultiplication of two Matrices : \n";

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

{

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

{

cout<<mat3[i][j]<<" ";

}

cout<<"\n";

}

int gd=DETECT,gm;

initgraph(&gd,&gm,NULL);

for(int i=0;i<n-1;i++)

{

line(b[i][0],b[i][1],b[i+1][0],b[i+1][1]);

}

line(b[2][0],b[2][1],b[0][0],b[0][1]);

for(int i=0;i<n-1;i++)

{

line(mat3[i][0],mat3[i][1],mat3[i+1][0],mat3[i+1][1]);

}

line(mat3[2][0],mat3[2][1],mat3[0][0],mat3[0][1]);

delay(5000);

closegraph();

}

};

int main()

{

matrix g;

int ch;

char ans;

g.get();

g.identitymat();

do

{

cout<<"menu\n1.translation\n2.scaling\n3.rotation";

cin>>ch;

switch(ch)

{

case 1:

g.trans();

g.multi();

g.display();

break;

case 2:

g.scale();

g.multi();

g.display();

break;

case 3:

g.rot();

g.multi();

g.display();

break;

}cin>>ans;

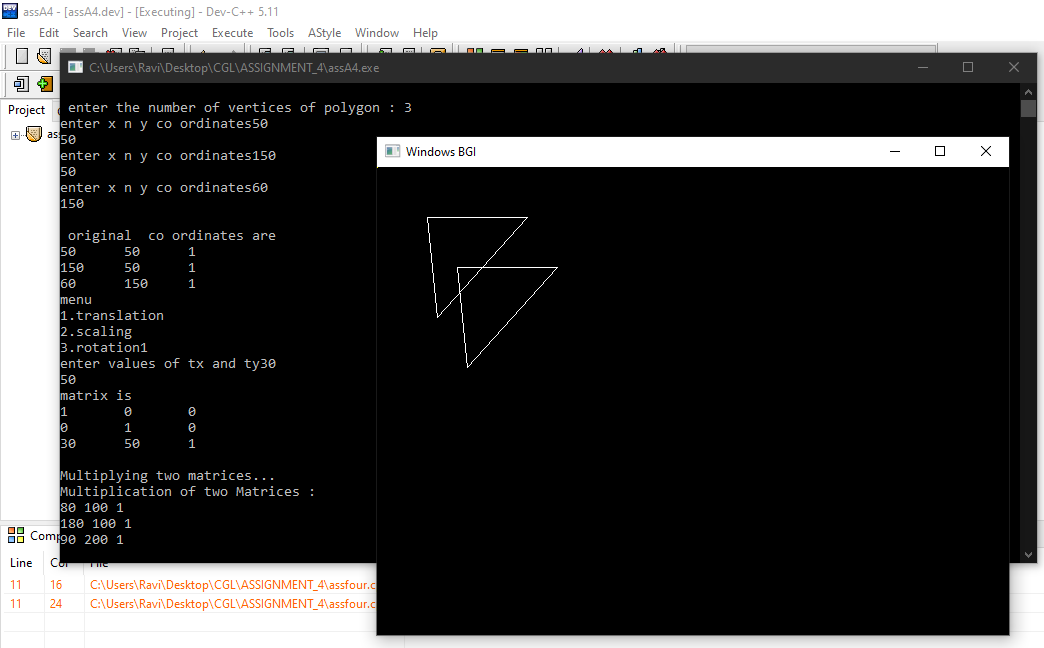
}while(ans=='Y'&& ans=='y');

return 0;

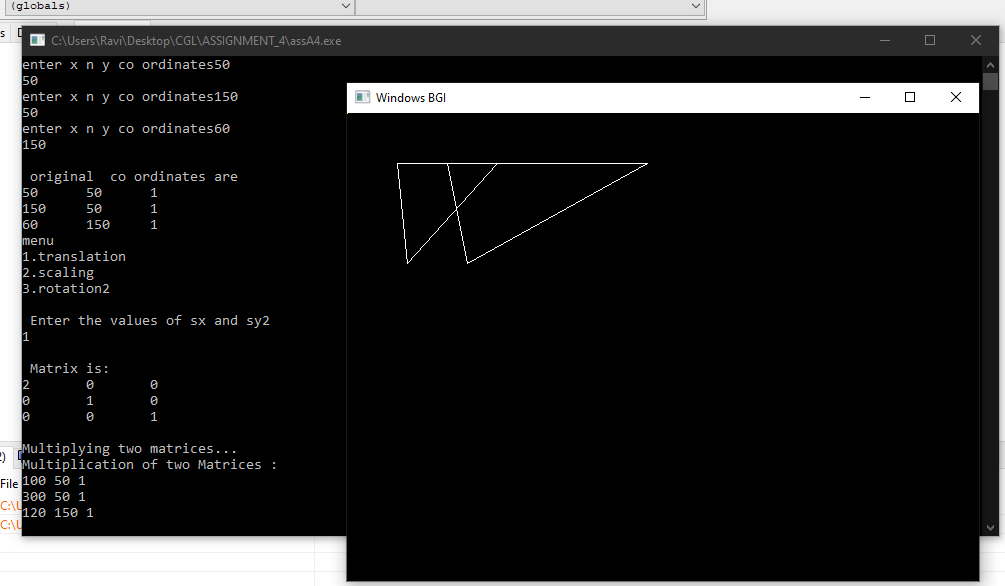
}

**OUTPUT :-**

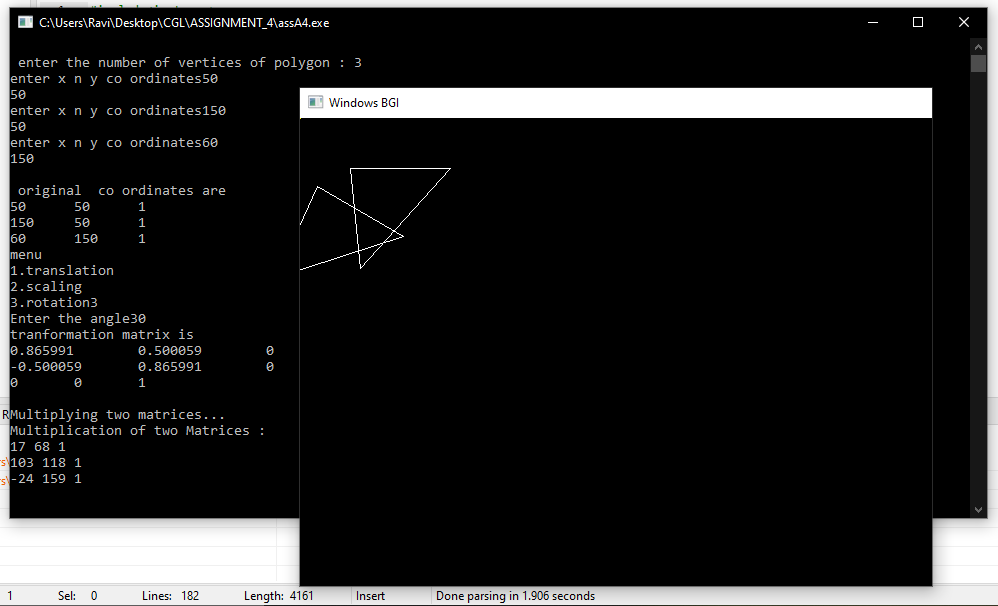
**1) Translation**

****

**2) Scaling**

****

**3) Rotation**

****