COMPUTER GRAPHICS PRACTICAL

**Q1: Write a program to clip a polygon using Sutherland Hodgeman algorithm.**

**Code:**

#include<iostream.h>

#include<conio.h>

#include<graphics.h>

int k,xmin,ymin,xmax,ymax,arr[20],m;

void clipl (int x1, int y1, int x2, int y2)

{

    if(x2-x1)

        m=(y2-y1)/(x2-x1);

    else

        m=100000;

    if(x1 >= xmin && x2 >= xmin)

    {

        arr[k]=x2;

        arr[k+1]=y2;

        k+=2;

    }

    if(x1 < xmin && x2 >= xmin)

    {

        arr[k]=xmin;

        arr[k+1]=y1+m\*(xmin-x1);

        arr[k+2]=x2;

        arr[k+3]=y2;

        k+=4;

    }

    if(x1 >= xmin  && x2 < xmin)

    {

        arr[k]=xmin;

        arr[k+1]=y1+m\*(xmin-x1);

        k+=2;

    }

}

void clipt(int x1, int y1, int x2, int y2)

{

    if(y2-y1)

        m=(x2-x1)/(y2-y1);

    else

        m=100000;

    if(y1 <= ymax && y2 <= ymax)

    {

        arr[k]=x2;

        arr[k+1]=y2;

        k+=2;

    }

    if(y1 > ymax && y2 <= ymax)

    {

        arr[k]=x1+m\*(ymax-y1);

        arr[k+1]=ymax;

        arr[k+2]=x2;

        arr[k+3]=y2;

        k+=4;

    }

    if(y1 <= ymax  && y2 > ymax)

    {

        arr[k]=x1+m\*(ymax-y1);

        arr[k+1]=ymax;

        k+=2;

    }

}

void clipr(int x1, int y1, int x2, int y2)

{

    if(x2-x1)

        m=(y2-y1)/(x2-x1);

    else

        m=100000;

    if(x1 <= xmax && x2 <= xmax)

    {

        arr[k]=x2;

        arr[k+1]=y2;

        k+=2;

    }

    if(x1 > xmax && x2 <= xmax)

    {

        arr[k]=xmax;

        arr[k+1]=y1+m\*(xmax-x1);

        arr[k+2]=x2;

        arr[k+3]=y2;

        k+=4;

    }

    if(x1 <= xmax  && x2 > xmax)

    {

        arr[k]=xmax;

        arr[k+1]=y1+m\*(xmax-x1);

        k+=2;

    }

}

void clipb(int x1, int y1, int x2, int y2)

{

    if(y2-y1)

        m=(x2-x1)/(y2-y1);

    else

        m=100000;

    if(y1 >= ymin && y2 >= ymin)

    {

        arr[k]=x2;

        arr[k+1]=y2;

        k+=2;

    }

    if(y1 < ymin && y2 >= ymin)

    {

        arr[k]=x1+m\*(ymin-y1);

        arr[k+1]=ymin;

        arr[k+2]=x2;

        arr[k+3]=y2;

        k+=4;

    }

    if(y1 >= ymin  && y2 < ymin)

    {

        arr[k]=x1+m\*(ymin-y1);

        arr[k+1]=ymin;

        k+=2;

    }

}

void main()

{

    int gd=DETECT,gm,n,poly[20];

    int xi,yi,xf,yf,polyy[20];

    initgraph(&gd,&gm,"C:\\TurboC3\\BGI");

    setcolor(WHITE);

    cout<<"Enter the Minimum Coordinates of visible window : x : ";

    cin>>xmin;

    cout<<"y : ";

    cin>>ymin;

    cout<<"Enter the Maximum Coordinates of visible window : x : ";

    cin>>xmax;

    cout<<"y : ";

    cin>>ymax;

    cout<<"Enter the number of side of Polygon to be clipped : ";

    cin>>n;

    cout<<"Enter the coordinates :";

    for(int i=0 ; i < 2\*n ; i++)

       cin>>polyy[i];

    polyy[i]=polyy[0];

    polyy[i+1]=polyy[1];

    for(i=0 ; i < 2\*n+2 ; i++)

poly[i]=polyy[i];

    clrscr();

    rectangle(xmin,ymax,xmax,ymin);

    cout<<"\tUNCLIPPED POLYGON";

    setcolor(WHITE);

    fillpoly(n,poly);

getch();

    clrscr();

    k=0;

    for(i=0;i < 2\*n;i+=2)

clipl(polyy[i],polyy[i+1],polyy[i+2],polyy[i+3]);

    n=k/2;

    for(i=0;i < k;i++)

polyy[i]=arr[i];

    polyy[i]=polyy[0];

    polyy[i+1]=polyy[1];

    k=0;

    for(i=0;i < 2\*n;i+=2)

clipt(polyy[i],polyy[i+1],polyy[i+2],polyy[i+3]);

    n=k/2;

    for(i=0;i < k;i++)

polyy[i]=arr[i];

    polyy[i]=polyy[0];

    polyy[i+1]=polyy[1];

    k=0;

    for(i=0;i < 2\*n;i+=2)

clipr(polyy[i],polyy[i+1],polyy[i+2],polyy[i+3]);

    n=k/2;

    for(i=0;i < k;i++)

polyy[i]=arr[i];

    polyy[i]=polyy[0];

    polyy[i+1]=polyy[1];

    k=0;

    for(i=0;i < 2\*n;i+=2)

clipb(polyy[i],polyy[i+1],polyy[i+2],polyy[i+3]);

    for(i=0;i < k;i++)

poly[i]=arr[i];

    if(k)

fillpoly(k/2,poly);

    setcolor(RED);

    rectangle(xmin,ymax,xmax,ymin);

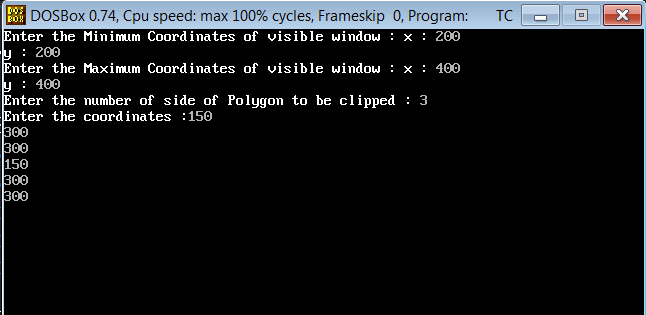
    cout<<"CLIPPED POLYGON";

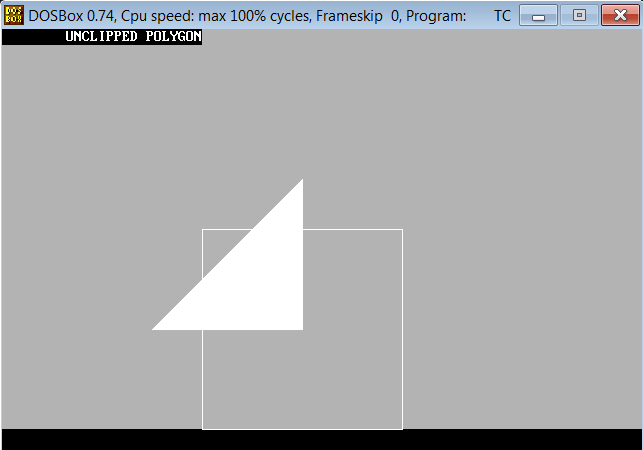
    getch();

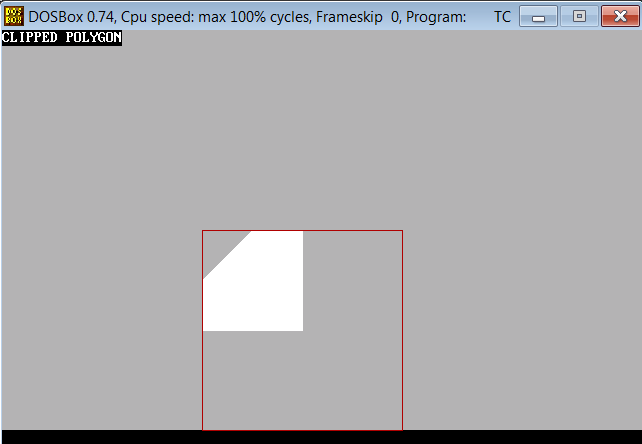
    closegraph();

}

**Output:**







**Q2: Write a program**

**Code:**

#include<iostream.h>

#include<graphics.h>

#include<conio.h>

#include<dos.h>

#include<math.h>

#define pi 3.14285714

class transformations

{

double vertices[3][3];             //matrix contains vertices of the triangle

double t\_matrix[3][3];             //transformation matrix

double result[3][3];

public:

transformations(){};

void get\_vertices();

void display\_triangle();

void display\_triangle\_result();

void multiplication();

void copyback();

void rotation(double angle,double m,double n);

void reflection(double m,double c);

void scaling(double a,double d);

void shearing(double b,double c);

};

void transformations::get\_vertices()

{

int i=0;

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

     {

cout<<"\nEnter vertex "<<i+1<<"...";

cout<<"\nx1 : ";

cin>>vertices[i][0];

result[i][0]=vertices[i][0];

cout<<"y1 : ";

cin>>vertices[i][1];

result[i][1]=vertices[i][1];

vertices[i][2]=result[i][2]=1;

     }

}

void transformations::display\_triangle()

{

int i=0;

for(i=0;i<2;i++)

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

     line(vertices[i][0],vertices[i][1],vertices[0][0],vertices[0][1]);

}

void transformations::display\_triangle\_result()

{

int i=0;

for(i=0;i<2;i++)

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

     line(result[i][0],result[i][1],result[0][0],result[0][1]);

}

void transformations::copyback()

{

int i=0,j=0;

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

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

result[i][j]=vertices[i][j];

}

void transformations::multiplication()

{

double r[3][3];

int i=0,j=0,k=0;

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

{

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

    {

r[i][j]=0;

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

r[i][j]+=result[i][k]\*t\_matrix[k][j];

    }

}

for(i=0;i<3;i++)              //Copying back the result

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

result[i][j]=r[i][j];

}

//REFLECTION:-

void transformations::reflection(double m,double c)

{

double angle=atan(m);                //tan inverse(slope)=angle

copyback();

cleardevice();

double x1=0,y1=c,x2=400,y2=(m\*x2)+c;

setcolor(YELLOW);

line(x1,y1,x2,y2);

delay(20);

getch();

setcolor(RED);

display\_triangle();

delay(20);

getch();

t\_matrix[0][0]=1;                     //translate (0,-c)

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

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

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

t\_matrix[1][1]=1;

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

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

t\_matrix[2][1]=(c\*(-1));

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

multiplication();

t\_matrix[0][0]=cos(-1\*angle);             //rotate the object about the x-axis by -angle

t\_matrix[0][1]=sin(-1\*angle);

t\_matrix[1][0]=(sin(-1\*angle)\*(-1));;

t\_matrix[1][1]=cos(-1\*angle);

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

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

multiplication();

t\_matrix[0][0]=1;                      //reflection about x-axis

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

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

t\_matrix[1][1]=-1;

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

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

multiplication();

t\_matrix[0][0]=cos(angle);             //rotate the object about the x-axis by -angle

t\_matrix[0][1]=sin(angle);

t\_matrix[1][0]=(sin(angle)\*(-1));;

t\_matrix[1][1]=cos(angle);

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

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

multiplication();

t\_matrix[0][0]=1;                       //translating back to original line of reflection

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

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

t\_matrix[1][1]=1;

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

t\_matrix[2][1]=c;

multiplication();

setcolor(GREEN);

display\_triangle\_result();

delay(20);

getch();

}

void main()

{

clrscr();

int gd=DETECT,gm,choice;

transformations t1;

char ch1,ch2;

double angle,m,n,slope,intercept,a,b,c,d;

do

    {

cout<<"\n\n\t   ........TWO DIMENSIONAL TRANSFORMATIONS........\n";

cout<<"\nEnter the details of a triangle(i.e. 2-D object).....";

       t1.get\_vertices();

do

  {

initgraph(&gd,&gm,"C:\\Turboc3\\BGI");

cout<<"\n.......MENU.......";

cout<<"\n1.Rotation.";

cout<<"\n2.Reflection.";

cout<<"\n3.Scaling.";

cout<<"\n4.Shearing.";

cout<<"\n..................";

cout<<"\n\nEnter your choice :: ";

cin>>choice;

switch(choice)

      {

case 1:cout<<"\n\nFOR ROTATION..........";

cout<<"\nEnter the angle of rotation :: ";

cin>>angle;

cout<<"\nNow,enter the point about which you wanna perform rotation :: ";

cout<<"\nx coordinate : ";

cin>>m;

cout<<"y coordinate : ";

cin>>n;

t1.rotation(angle,m,n);

break;

case 2:cout<<"\n\nFOR REFLECTION..........";

cout<<"\nTo enter the line in slope-intercept form(i.e. y=mx+c)....";

cout<<"\nEnter slope(m) : ";

cin>>slope;

cout<<"Then,enter y-intercept(c) : ";

cin>>intercept;

t1.reflection(slope,intercept);

break;

case 3:cout<<"\n\nFOR SCALING..........";

cout<<"\nEnter the factor of scaling...";

cout<<"\nAlong the x-axis : ";

cin>>a;

cout<<"And, along the y-axis : ";

cin>>d;

t1.scaling(a,d);

break;

case 4:cout<<"\n\nFOR SHEARING..........";

cout<<"\nEnter the factor of shearing...";

cout<<"\nAlong the x-axis : ";

cin>>c;

cout<<"And, along the y-axis : ";

cin>>b;

t1.shearing(b,c);

break;

default:cout<<"\n\n\t!!!INVALID CHOICE!!!";

getch();

      }

closegraph();

cout<<"\nWanna try another tarnsformation(y/n)...";

cin>>ch2;

}while(ch2=='y');

cout<<"\n\nWant to try with a triangle of different dimensions(y/n)? ";

cin>>ch1;

}while(ch1=='y');

}

**Output:**

