**COHEN SUTHERLAND LINE CLIPPING:**

Code:

#include<iostream.h>

#include<graphics.h>

#include<conio.h>

/\*TOP=0001, LEFT=1000, RIGHT=0100, BOTTOM=0010\*/

int compOutCode(

double x,double y,

double xmin,double xmax,

double ymin,double ymax)

{

int code=0000;

if(y>ymax)

code+=0001;

else if(y<ymin)

code+=0010;

if(x>xmax)

code+=0100;

else if(x<xmin)

code+=1000;

return code;

}

void cohen(

double x0,double y0,

double x1, double y1,

double xmin,double xmax,

double ymin,double ymax)

{

cout<<"Applying Cohen-Sutherlan Algorithm";

int outcode0,outcode1,outcodeout;

int acc=0,done=0;

outcode0=compOutCode(x0,y0,xmin,xmax,ymin,ymax);

outcode1=compOutCode(x1,y1,xmin,xmax,ymin,ymax);

do

{

if(!(outcode0|outcode1))

{

acc=1;done=1;

}

else if(outcode0&outcode1)

{ done=1;

}

else{

double x,y;

outcodeout=outcode0?outcode0:outcode1;

if(outcodeout&0001)

{

x=x0+(x1-x0)\*(ymax-y0)/(y1-y0);

y=ymax;

cout<<"\nClipping Top!!!";

}

else if(outcodeout&0010)

{

x=x0+(x1-x0)\*(ymin-y0)/(y1-y0);

y=ymin;

cout<<"\nClipping Bottom!!!";

}

else if(outcodeout&0100)

{y=y0+(y1-y0)\*(xmax-x0)/(x1-x0);

x=xmax;

cout<<"\nClipping Right";

}

else

{y=y0+(y1-y0)\*(xmin-x0)/(x1-x0);

x=xmin;

cout<<"\nClipping Left";

}

if(outcodeout==outcode0)

{

x0=x;y0=y;outcode0=compOutCode(x0,y0,xmin,xmax,ymin,ymax);

}

else

{

x1=x;y1=y;outcode1=compOutCode(x1,y1,xmin,xmax,ymin,ymax);

}

}

}

while(done==0);

clrscr();

cout<<"After clipping window and line:";

rectangle(xmin,ymax,xmax,ymin);

line(x0,y0,x1,y1);

getch();

return;

}

void main()

{

int i,gd=DETECT,gm;

int l,r,b,t;

initgraph(&gd,&gm,"C:\\TURBOC3\\BGI");

cout<<"Enter the coordinates of clipping window:\n";

cout<<"left: ";

cin>>l;

cout<<"bottom: ";

cin>>b;

cout<<"right: ";

cin>>r;

cout<<"top: ";

cin>>t;

int x0,x1,y0,y1;

cout<<"Enter the coordinates of original line:\n";

cout<<"x0: ";

cin>>x0;

cout<<"y0: ";

cin>>y0;

cout<<"x1: ";

cin>>x1;

cout<<"y1: ";

cin>>y1;

clrscr();

cout<<"Before clipping window and line:";

rectangle(l,t,r,b);

line(x0,y0,x1,y1);

getch();

cohen(x0,y0,x1,y1,l,r,b,t);

return;

}

output: //output:left-100,bottom-200,right-300,top-400 line:(200,250),(100,150)

**2-D rotation about arbitrary point**

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 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];

}

//ROTATION:-

void transformations::rotation(double angle,double m,double n)

{

angle=((pi/180)\*angle);

copyback();

cleardevice();

setcolor(RED);

display\_triangle();

delay(20);

getch();

t\_matrix[0][0]=1; //Matrix for translating the triangle to origin

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]=(m\*(-1));

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

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

multiplication();

t\_matrix[0][0]=cos(angle); //Matrix to perform rotation about origin

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; //Matrix to translate back to original centre of rotation

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

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

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

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

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

multiplication();

setcolor(GREEN);

display\_triangle\_result();

delay(20);

getch();

}

void main()

{

clrscr();

int gd=DETECT,gm;

transformations t1;

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

{

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

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

t1.get\_vertices();

{

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

{

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);

getch();

}

closegraph();

}}}

Output: //output coordinates: (100,200),(400,200),(500,100) angle:45