**COMPUTER GRAPHICS**

**LAB FILE**

****

**AMITY SCHOOL OF ENGINEERING AND TECHNOLOGY**

**AMITY UNIVERSITY**

**UTTAR PRADESH**

**SUBMITTED BY: SUBMITTED TO:**

**Index**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **S.No** | **Category of Assignment** | **Code** | **Name of Experiment** | **Date of Allotment of experiment** | **Date of Evaluation** | **Max.**  **Marks** | **Marks obtained** | **Sign. of Faculty** |
|  | **Mandatory Experiment** | **LR (10)** | WAP to draw a line using DDA algorithms | 20/12/16 |  | **1** |  |  |
|  |  | WAP to draw a line using Bresenham's algorithms | 3/01/17 |  | **1** |  |  |
|  |  | WAP to draw a circle using Bresenham's algorithms | 10/01/17 |  | **1** |  |  |
|  |  | WAP to draw a circle using Mid-point algorithms | 17/01/17 |  | **1** |  |  |
|  |  | WAP to draw a ellipse using Mid-point algorithms | 31/01/17 |  | **1** |  |  |
|  | Cohen Sutherland clipping algorithm | 07/02/17 |  | **1** |  |  |
|  |  | WAP to translate and scale a triangle | 14/02/17 |  | **1** |  |  |
| 1. **adsf** |  | WAP to rotate a triangle | 28/02/17 |  | **1** |  |  |
|  |  | WAP to reflect a triangle | 07/03/17 |  | **1** |  |  |
|  |  | | | | | | | |
|  | **OPEN ENDED EXPERIMENT** | **pr**  **(10)** | Wap to draw a hyperbola |  |  | **10** |  |  |

**EXPERIMENT 1**

**DATE:**

**OBJECTIVE:** Introduction graphics programming

**SOFTWARE USED:** TurboC3

**SOURCE CODE**

#include<stdio.h>

#include<graphics.h>

#include<conio.h>

void main(){

int gd = DETECT,gm;

int x ,y ,radius=80;

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

x = getmaxx()/2;

y = getmaxy()/2;

outtextxy(x-100, 50, "Circle and Line Using Inbuilt Functions");

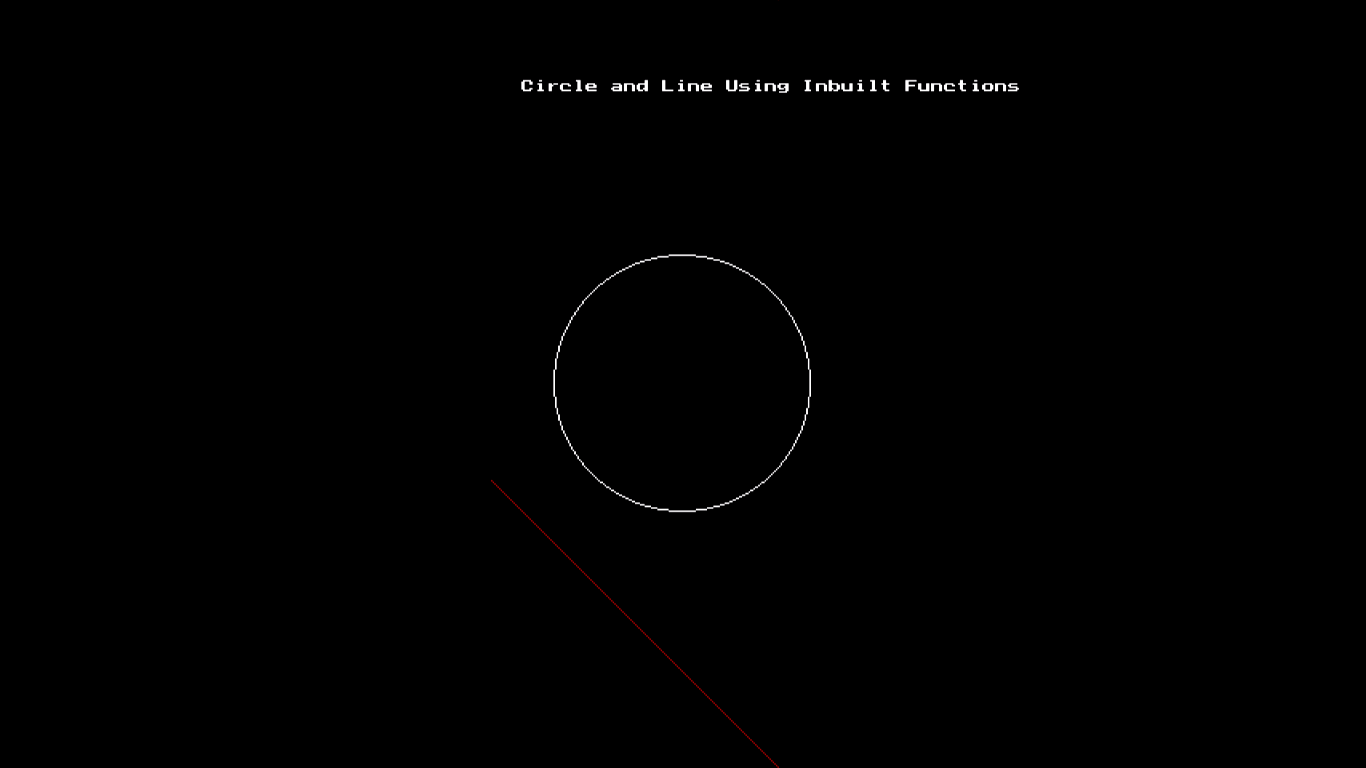
circle(x, y, radius);

setcolor(RED);

line(200,300,400,500);

getch(); }

**OUTPUT:**



**CONCLUSIONS:** The line and circle is plotted on the screen using inbuilt functions

**SIGNATURE:**

**Experiment 2**

**DATE:**

**OBJECTIVE:** To draw a line using DDA algorithm

**SOFTWARE USED:** Turbo C3

**Source Code :**

include<iostream.h>

#include<conio.h>

#include<stdio.h>

#include<graphics.h>

#include<dos.h>

#include<stdlib.h>

void drawline(float,float,float,float);

void main(){clrscr();

int gdriver = DETECT, gmode;

initgraph(&gdriver, &gmode, "C:\\TC\\bgi");

drawline(100,200,100,300);

drawline(100,200,150,250);

drawline(150,250,100,300);

getch();}

void drawline(float x1,float y1,float x2,float y2){

float dx,dy,x,y,length;

int i;

dx=x2-x1;

dy=y2-y1;

if(dx>dy) {length=dx;}

else {length=dy;}

dx=(x2-x1)/length;

dy=(y2-y1)/length;

//cout<<"dx and dy are "<<dx<<" and "<<dy;

//getch();

x=x1;y=y1;

for(i=1;i<=length;i++){

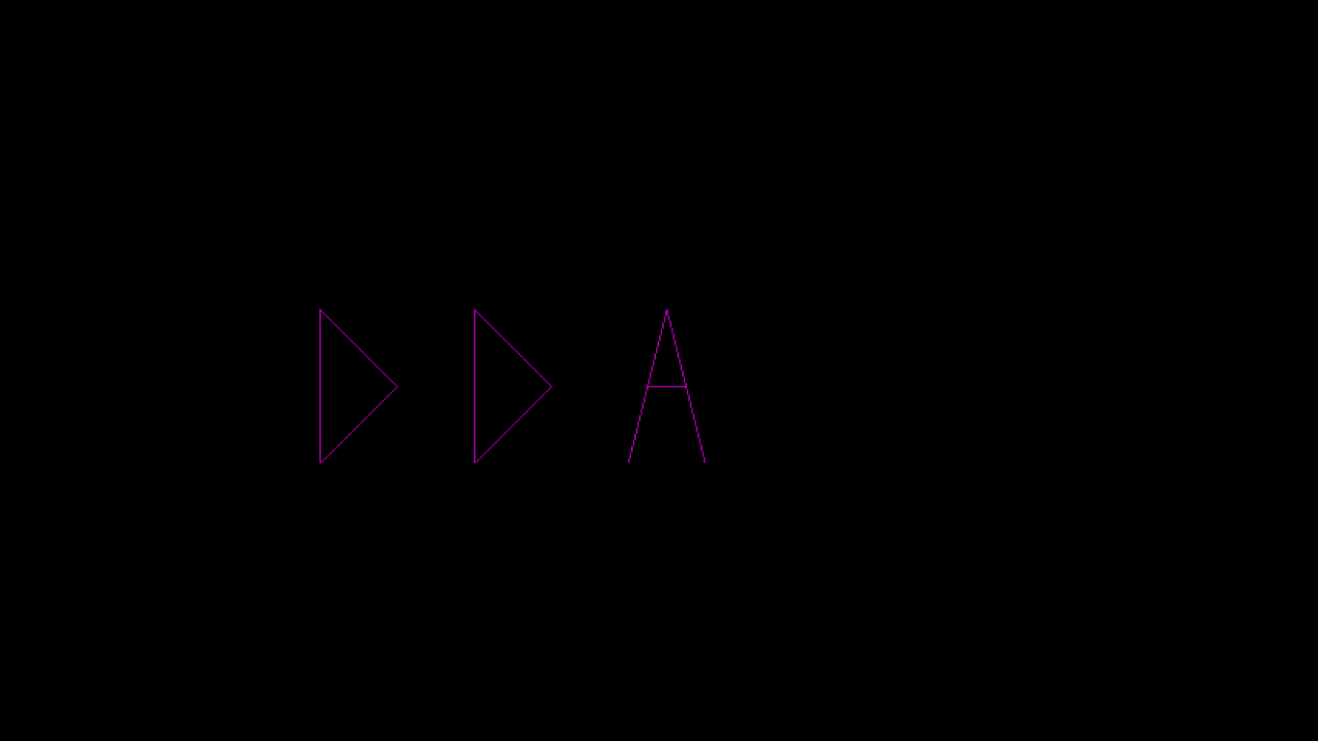
putpixel(x,y,MAGENTA);

x=x+dx;

y=y+dy;

//delay(50); }}

**OUTPUT:**



**CONCLUSIONS:** The line is plotted on the screen using DDA Algorithm.

**SIGNATURE:**

**EXPERIMENT 3**

**DATE:**

**OBJECTIVE:** To draw a line using Bresenham’s line drawing algorithm

**SOFTWARE USED:** Turbo C3

**PRACTICAL:**

#include<iostream.h>

#include<graphics.h>

#include<conio.h>

#include<math.h>

void main()

{

clrscr();

int gd=DETECT,gm;

int x1,y1,x2,y2,m,Pk,dx,dy;

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

cout<<" Enter initial coordinates:\n \n”;

cin>>x1>>y1;

cout<<" Enter final coordinates:\n \n”;

cin>>x2>>y2;

dx=x2-x1;

dy=y2-y1;

m=dy/dx;

if(m<1)

{

Pk=2\*dy-dx;

while(x1!=x2)

{

if(Pk<0)

{

x1+=1;

putpixel(x1,y1,WHITE);

Pk=Pk+2\*dy;

}

else

{

x1+=1;

y1+=1;

putpixel(x1,y1,WHITE);

Pk=Pk+2\*dy-2\*dx; } } }

else

{

Pk=2\*dx-dy;

while(y1!=y2)

{

if(Pk<0)

{

y1+=1;

putpixel(x1,y1,WHITE);

Pk=Pk+2\*dx;

}

else

{

x1+=1;

y1+=1;

putpixel(x1,y1,WHITE);

Pk=Pk+2\*dx-2\*dy; } }

}

getch();

closegraph();

}

**RESULT:**



**CONCLUSIONS: L**ine is plotted on the screen using Bresenham’s Algorithm.

**SIGNATURE:**

**EXPERIMENT 4**

**DATE:**

**OBJECTIVE:** To draw a circle using Bresenham’s Circle drawing algorithm

**SOFTWARE USED:** Turbo C3

**PRACTICAL:**

#include<iostream.h>

#include<conio.h>

#include<graphics.h>

void main()

{

int gd=DETECT,gm;

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

float Pk,Xc,Yc,r,x,y;

cout<<"Enter radius of circle\n”;

cin>>r;

cout<<"Enter centre coordinates of circle\n”;

cin>>Xc>>Yc;

x=0,y=r;

Pk=3-2\*r;

while(x<=y)

{

if(Pk<0)

{

x++;

Pk=Pk+4\*x+6;

}

else

{

x++;

y--;

Pk=Pk+4\*x-4\*y+10;

}

putpixel(x+Xc,y+Yc,WHITE);

putpixel(-x+Xc,y+Yc,WHITE);

putpixel(y+Xc,x+Yc,WHITE);

putpixel(-y+Xc,x+Yc,WHITE);

putpixel(y+Xc,-x+Yc,WHITE);

putpixel(-y+Xc,-x+Yc,WHITE);

putpixel(x+Xc,-y+Yc,WHITE);

putpixel(-x+Xc,-y+Yc,WHITE);

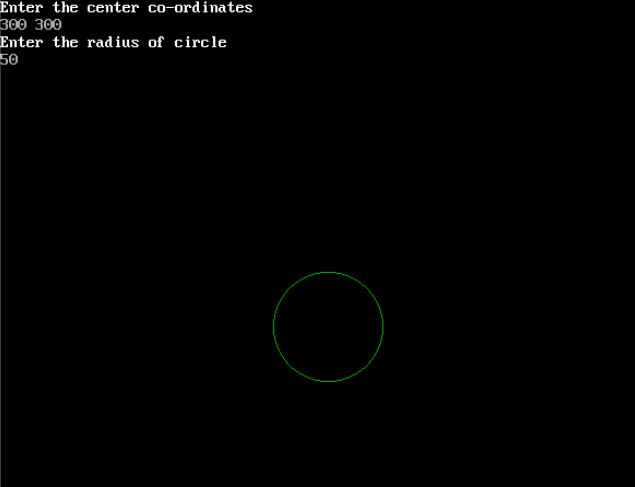
}

getch();

closegraph();

}

**RESULT:**



**CONCLUSIONS:** A circle is plotted on the screen using Bresenham’s Algorithm.

**SIGNATURE:**

**EXPERIMENT 5**

**DATE:**

**OBJECTIVE:** To draw a circle using Midpoint Circle drawing algorithm

**SOFTWARE USED:** Turbo C++

**PRACTICAL:**

#include<iostream.h>

#include<conio.h>

#include<graphics.h>

void main()

{

clrscr();

int gd=DETECT,gm;

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

float Pk,Xc,Yc,r,x,y;

cout<<"Enter radius of the circle\n”;

cin>>r;

cout<<"Enter centre coordinates of the circle\n”;

cin>>Xc>>Yc;

x=0,y=r;

Pk=(5/4)-r;

while(x<=y)

{

if(Pk<0)

{

x++;

Pk=Pk+2\*x+3;

}

else

{

x++;

y--;

Pk=Pk+2\*x-2\*y+5;

}

putpixel(x+Xc,y+Yc,WHITE);

putpixel(-x+Xc,y+Yc,WHITE);

putpixel(y+Xc,x+Yc,WHITE);

putpixel(-y+Xc,x+Yc,WHITE);

putpixel(y+Xc,-x+Yc,WHITE);

putpixel(-y+Xc,-x+Yc,WHITE);

putpixel(x+Xc,-y+Yc,WHITE);

putpixel(-x+Xc,-y+Yc,WHITE);

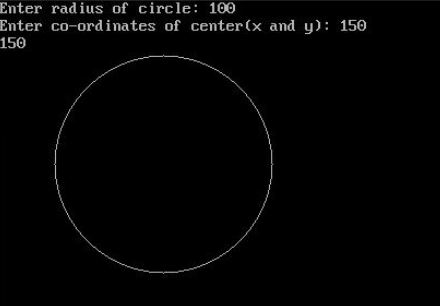
}

getch();

closegraph();

}

**RESULT:**



**CONCLUSIONS** –A circle is plotted on the screen using Midpoint Algorithm.

**SIGNATURE:**

**EXPERIMENT 6**

**DATE:**

**OBJECTIVE:** To draw an Ellipse using midpoint algorithm

**SOFTWARE USED:** Turbo C++

**SOURCE CODE:**

#include<iostream.h>

#include<conio.h>

#include<math.h>

#include<graphics.h>

void main()

{

clrscr();

int gd=DETECT,gm;

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

float x,y,rx,ry,Xc,Yc;

float pk;

cout<<"Enter major axis\n”;

cin>>rx;

cout<<"Enter minor axis\n”;

cin>>ry;

cout<<"Enter centre\n”;

cin>>Xc>>Yc;

x=0,y=ry;

float Rx=pow(rx,2),Ry=pow(ry,2);

pk=Ry-(Rx\*ry)+(Rx/4);

while((2\*Ry\*x)<=(2\*Rx\*y))

{

if(pk<0)

{

x=x+1;

pk=pk+(2\*Ry\*x)+Ry;

}

else

{

x=x+1;

y=y-1;

pk=pk+(2\*Ry\*x)-(2\*Rx\*y)+Ry;

}

putpixel(x+Xc,y+Yc,WHITE);

putpixel(-x+Xc,y+Yc,WHITE);

putpixel(x+Xc,-y+Yc,WHITE);

putpixel(-x+Xc,-y+Yc,WHITE);

}

pk=(Ry\*pow((x+0.5),2))+(Rx\*pow((y-1),2))-(Rx\*Ry);

while((2\*Rx\*y)!=0)

{

if(pk<0)

{

x=x+1;

y=y-1;

pk=pk-(2\*Rx\*y)+(2\*Ry\*x)+Rx;

}

else

{

y=y-1;

pk=pk-(2\*Rx\*y)+Rx;

}

putpixel(x+Xc,y+Yc,WHITE);

putpixel(-x+Xc,y+Yc,WHITE);

putpixel(x+Xc,-y+Yc,WHITE);

putpixel(-x+Xc,-y+Yc,WHITE);

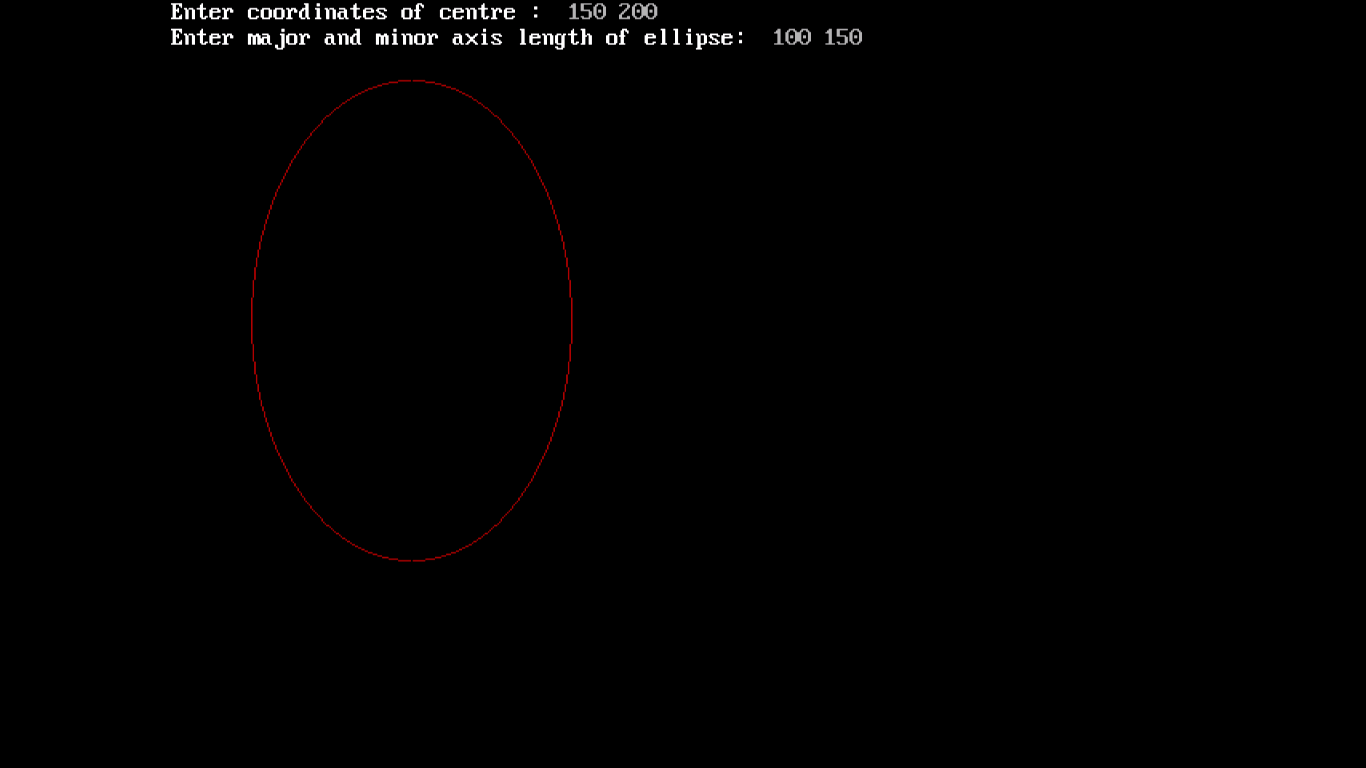
}

getch();

closegraph();

}

**RESULT:**



**CONCLUSIONS:** An ellipse is plotted on the screen using midpoint Algorithm.

**SIGNATURE:**

**EXPERIMENT 7**

**DATE:**

**OBJECTIVE:** WAP to translate and scale a polygon

**SOFTWARE USED:** TURBO C3

**SOURCE CODE:**

#include <graphics.h>

#include <stdlib.h>

#include <stdio.h>

#include <conio.h>

#include<math.h>

int x1,y1,x[20],y[20],n;

float theta;

void draw\_poly()

{

int i;

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

{

line(x[i],y[i],x[i+1],y[i+1]);

}

line(x[0],y[0],x[n-1],y[n-1]);

}

void translate(int x1,int y1)

{

int a;

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

{

x[a]=x[a]+x1;

y[a]=y[a]+y1;

}

}

void scale(int x1,int y1)

{

int a,b;

a=x[0];

b=y[0];

translate(-a,-b);

int i;

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

{

x[i]=x[i]\*x1;

y[i]=y[i]\*y1;

}

translate(a,b);

}

void main()

{

int gdriver = DETECT, gmode;

initgraph(&gdriver, &gmode, “”);

int i;

clrscr();

cout<<"\nEnter no. of Sides:: ";

cin>>n;

cout<<"\nEnter the Co-ordinate of Vertices::";

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

{

cout<<"\n"<<i+1<<"::";

cin>>x[i]>>y[i];

}

draw\_poly();

getch();

cleardevice();

clrscr();

int k;

do

{

clrscr();

cleardevice();

cout<<"\n1)Translate\n2)Scale\n3)Exit.";

cout<<"\nEnter Ur Choice::";

cin>>k;

switch(k)

{

case 1:

cout<<"\nFor x-axis::";

cin>>x1;

cout<<"\nFor y-axis::";

cin>>y1;

translate(x1,y1);

draw\_poly();

getch();

cleardevice();

break;

case 2:

cout<<"\nFor x-axis::";

cin>>x1;

cout<<"\nFor y-axis::";

cin>>y1;

scale(x1,y1);

draw\_poly();

getch();

cleardevice();

break;

}

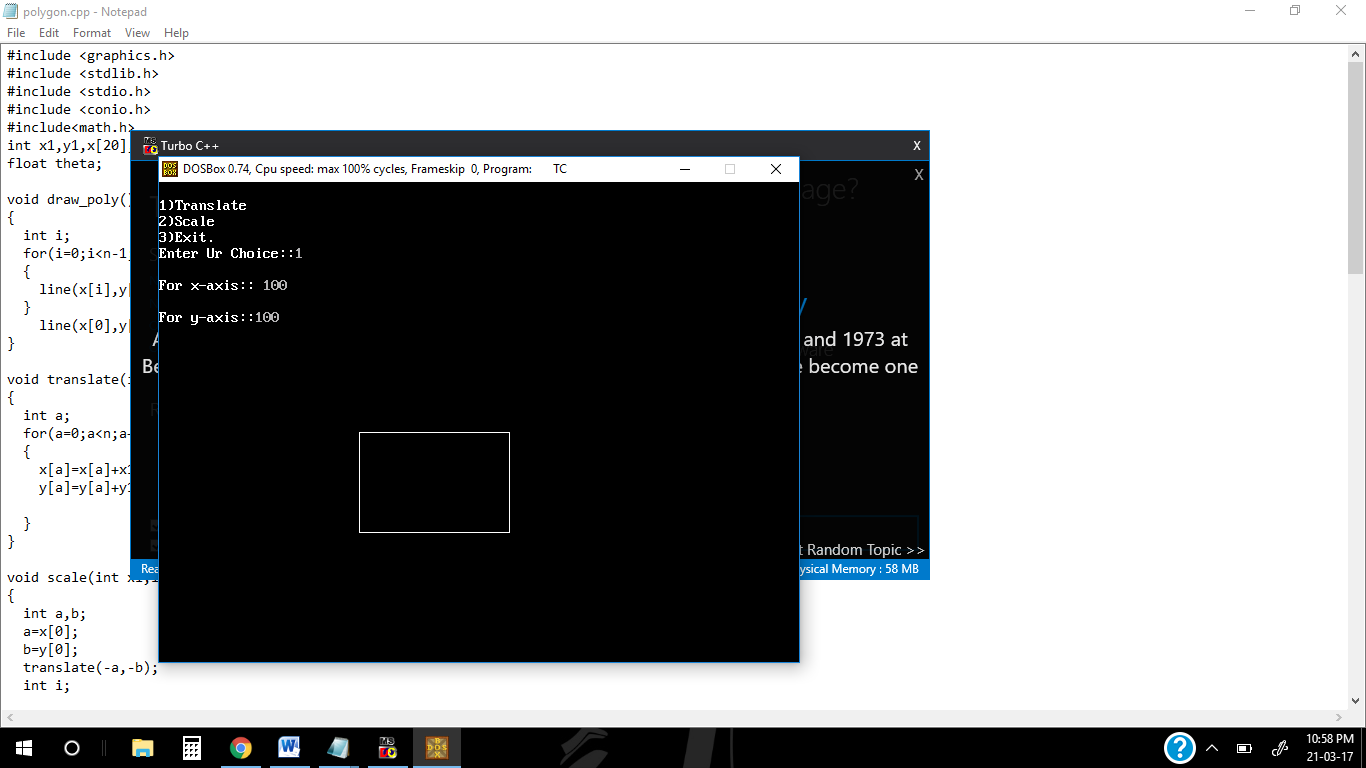
} while(k!=3);

return 0;

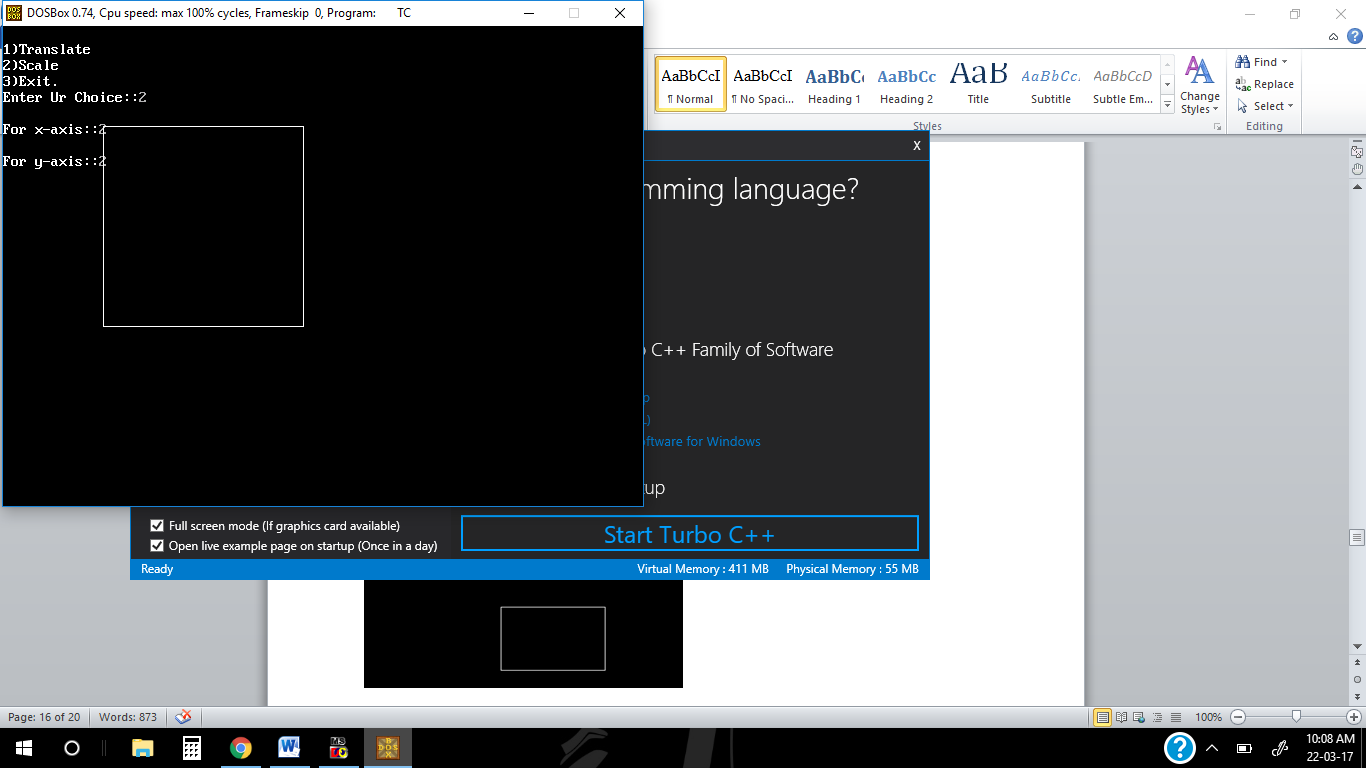
}



TRANSLATE:



SCALING:



**CONCLUSIONS:** Scaling and translation of a polygon is plotted on screen using matrix multiplication.

**SIGNATURE:**

**EXPERIMENT 8**

**DATE:**

**OBJECTIVE:** WAP to rotate a polygon

**SOFTWARE USED:** TURBO C3

**SOURCE CODE:**

#include<stdio.h>

#include<graphics.h>

#include<math.h>

int graDriver=DETECT,graMode;

int n,xs[100],ys[100],i,xp,yp,degree;

float radian;

void rotation();

void DrawFn();

void degToRad()

{

radian=(float)degree\*3.14f/180;

}

void main()

{

printf("Enter number of sides: ");

scanf("%d",&n);

printf("Enter co-rdinates: x,y for each point ");

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

scanf("%d%d",&xs[i],&ys[i]);

printf("\nenter pivot point co-ordinate");

scanf("%d%d",&xp,&yp);

printf("\nenter rotation angle");

scanf("%d",&degree);

degToRad();

initgraph(&graDriver,&graMode,"");

cleardevice();

//Drawing original in RED color

setcolor(RED);

DrawFn();

//Doing rotation

rotation();

//Drawing rotated polygon in BLUE color

setcolor(BLUE);

DrawFn();

getch();

}

void DrawFn()

{

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

line(xs[i],ys[i],xs[(i+1)%n],ys[(i+1)%n]);

}

void rotation()

{

float t,v;

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

{

t=xs[i]-xp;

v=ys[i]-yp;

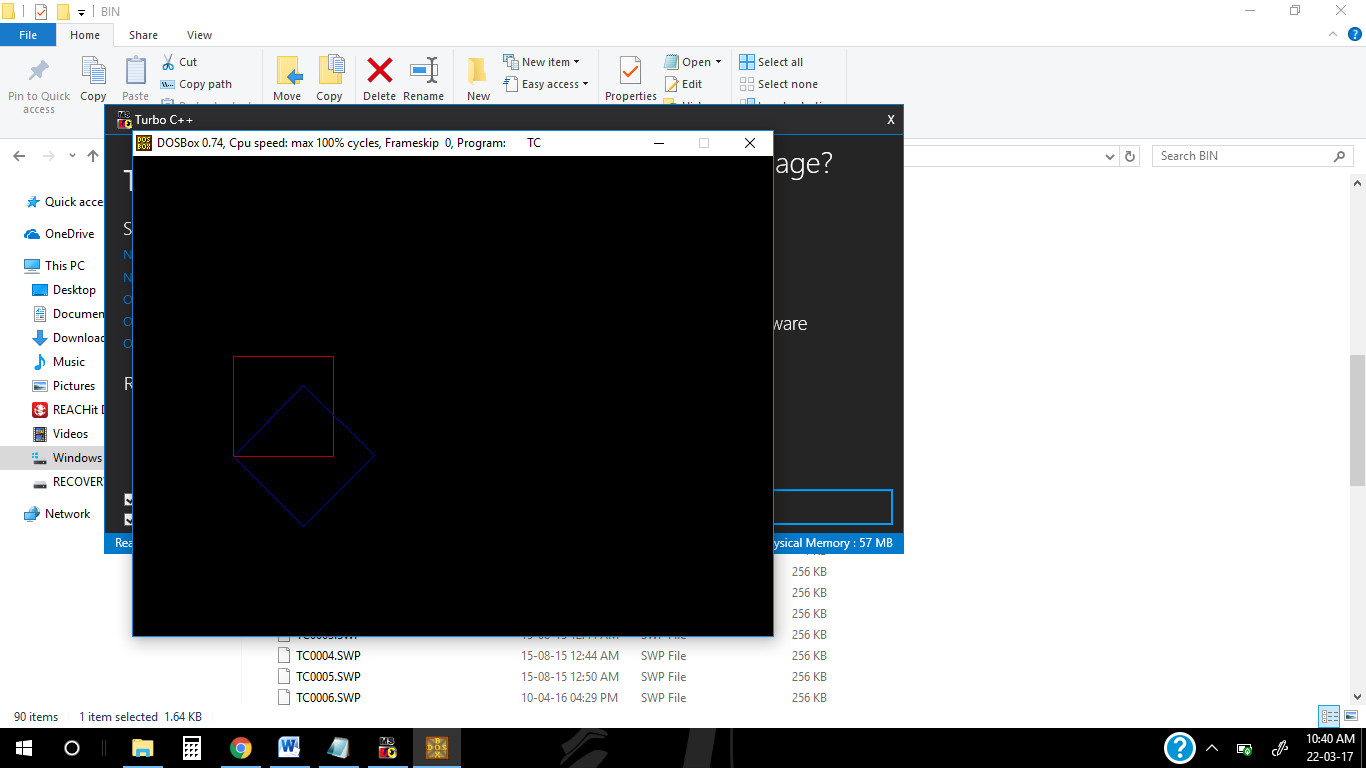
xs[i]=xp+floor(t\*cos(radian)-v\*sin(radian));

ys[i]=yp+floor(v\*cos(radian)+t\*sin(radian));

}

}

OUTPUT:



**CONCLUSIONS:** Rotation of a polygon is plotted on screen using matrix multiplication.

**SIGNATURE:**

**EXPERIMENT 9**

**DATE:**

**OBJECTIVE:** WAP to reflect a triangle

**SOFTWARE USED:** TURBO C3

**SOURCE CODE:**

#include<stdio.h>

#include<conio.h>

#include<graphics.h>

void main()

{

int a,a1,b,b1,c,c1,xt,ch;

int gd=DETECT,gm;

initgraph(&gd,&gm,"");

a=getmaxx();

a1=getmaxy();

b=a/2;

b1=a1/2;

line(b,0,b,a1);

line(0,b1,a,b1);

line(400,200,600,200);

line(400,200,400,100);

line(400,100,600,200);

printf("1.x-axis\n");

printf("2.y-axis\n");

printf("3.exit\n");

do

{

printf("Enter your choice\n");

scanf("%d",&ch);

switch(ch)

{

case 1:

c=400-b;c1=200-b1;

line(b+c,b1-c1,b+c+200,b1-c1);

line(b+c,b1-c1,b+c,b1-c1+100);

line(b+c,b1-c1+100,b+c+200,b1-c1);

break;

case 2:

c=400-b;c1=200-b1;

line(b-c,b1+c1,b-c-200,b1+c1);

line(b-c,b1+c1,b-c,b1+c1-100);

line(b-c,b1+c1-100,b-c-200,b1+c1);

break;

}

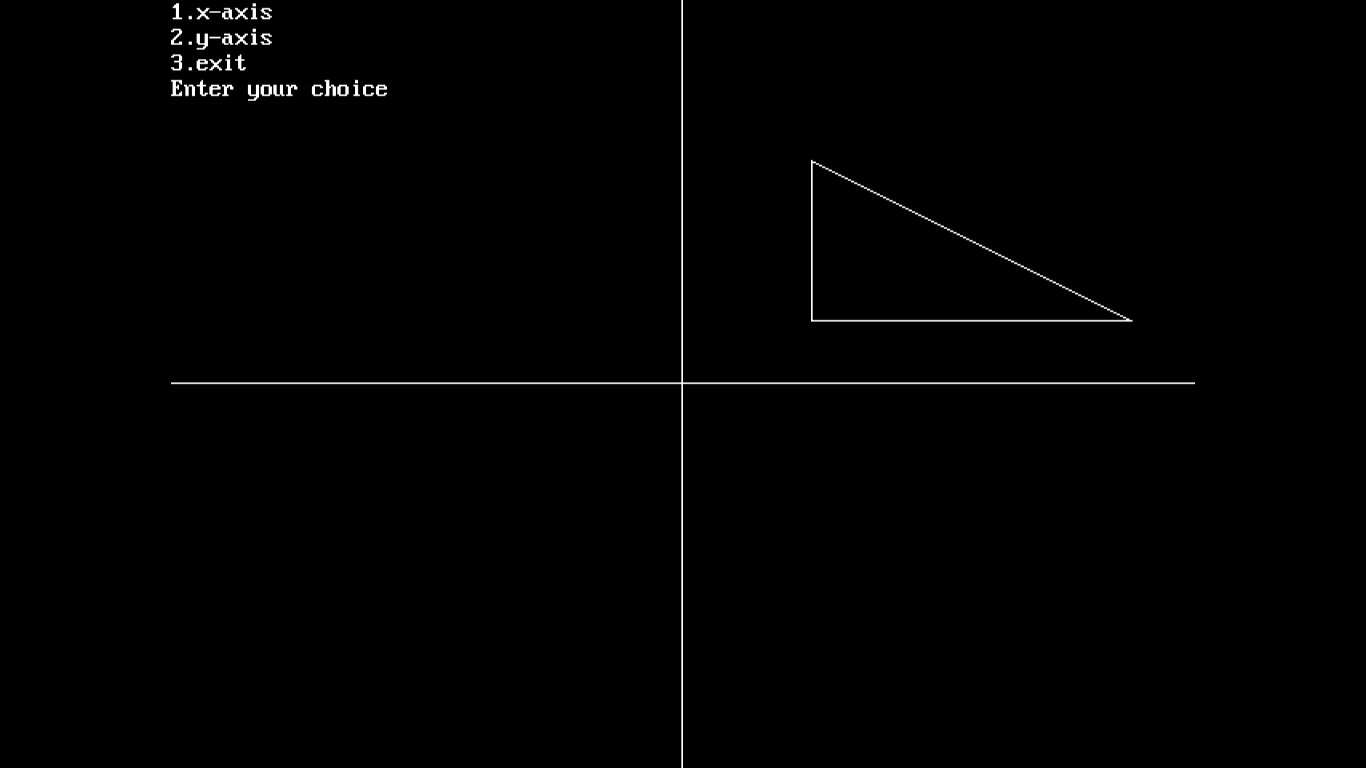
}while(ch<3);

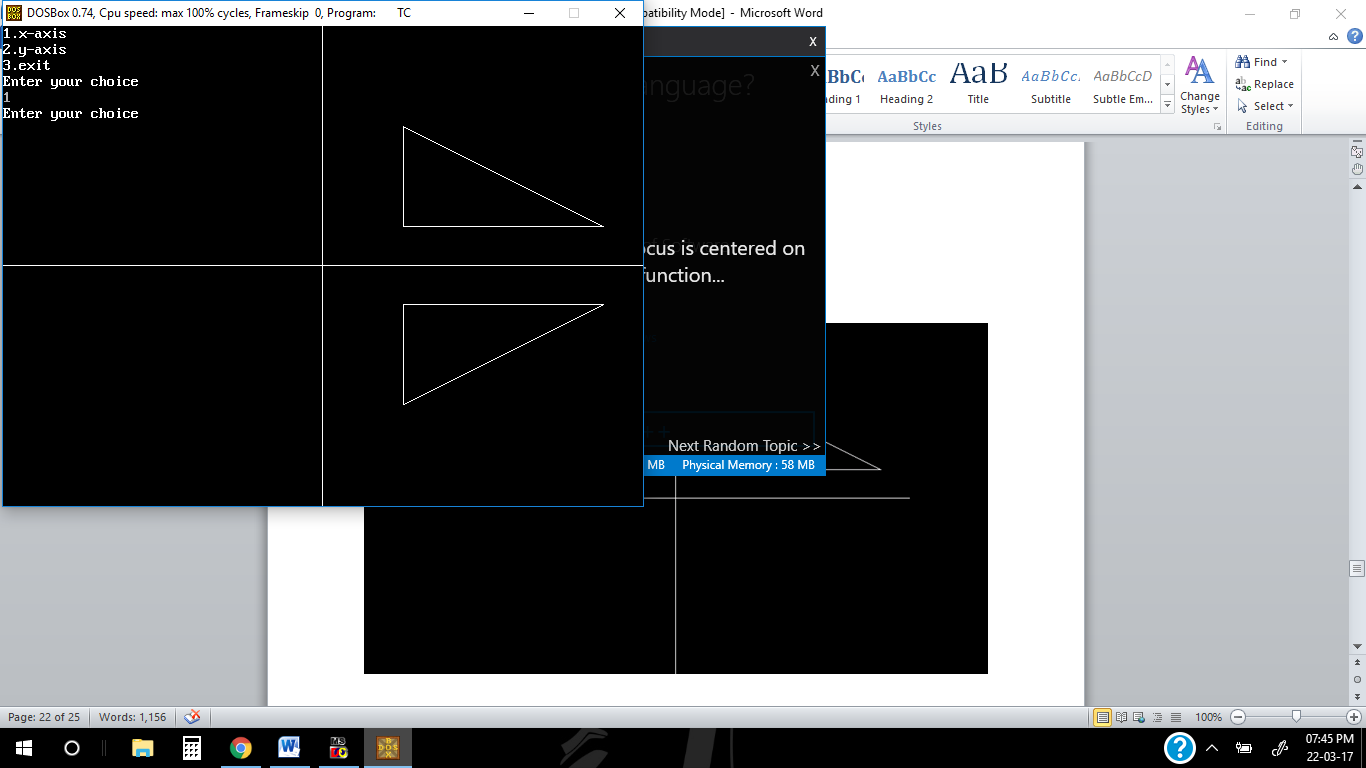
getch();

closegraph();

}

OUTPUT:





**CONCLUSIONS:** Reflection about x-axis and y-axis is plotted on screen using matrix multiplication.

**SIGNATURE:**

**EXPERIMENT 10**

**DATE:**

**OBJECTIVE:** To implement Cohen Sutherland line clipping algorithm

**SOFTWARE USED:** Turbo C3

**SOURCE CODE:**

#include <stdio.h>

#include <graphics.h>

#include <conio.h>

#include <math.h>

#define TRUE 1

#define FALSE 0

typedef unsigned intoutcode;

outcodeCompOutCode(float x,float y);

enum

{

TOP = 0x1,

BOTTOM = 0x2,

RIGHT = 0x4,

LEFT = 0x8

};

floatxmin,xmax,ymin,ymax;

void clip(float x0,float y0,float x1,float y1)

{

outcode outcode0,outcode1,outcodeOut;

int accept = FALSE,done = FALSE;

outcode0 = CompOutCode(x0,y0);

outcode1 = CompOutCode(x1,y1);

do

{

if(!(outcode0|outcode1))

{

accept = TRUE;

done = TRUE;

}

else if(outcode0 & outcode1)

done = TRUE;

else

{

float x,y;

outcodeOut = outcode0?outcode0:outcode1;

if (outcodeOut& TOP)

{

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

y = ymax;

}

else if (outcodeOut& BOTTOM)

{

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

y = ymin;

}

else if (outcodeOut& RIGHT)

{

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

x = xmax;

}

else

{

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

x = xmin;

}

if (outcodeOut==outcode0)

{

x0 = x;

y0 = y;

outcode0 = CompOutCode(x0,y0);

}

}while(done==FALSE);

if(accept)

line(x0,y0,x1,y1);

outtextxy(200,20,"LINE AFTER CLIPPING");

rectangle(xmin,ymin,xmax,ymax);

}

outcodeCompOutCode(float x,float y)

{

outcode code = 0;

if(y>ymax)

code|=TOP;

else

if(y<ymin)

code|=BOTTOM;

if(x>xmax)

code|=RIGHT;

else

if(x<xmin)

code|=LEFT;

return code;

}

void main( )

{

float x1,y1,x2,y2;

int gdriver = DETECT, gmode ;

printf("\nEnter the endpoints of line\n");

scanf("%f%f%f%f",&x1,&y1,&x2,&y2);

printf("Enter the rectangular coordinates of clipping window\n");

scanf("%f%f%f%f",&xmin,&ymin,&xmax,&ymax);

/\* initialize graphics and local variables \*/

initgraph(&gdriver, &gmode, "c:\\tc\\bgi");

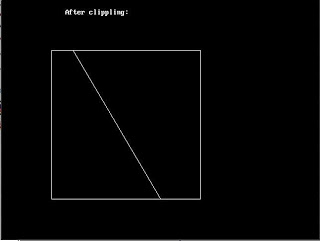
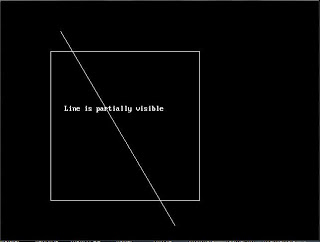
outtextxy(200,20,"LINE BEFORE CLIPPING");

line(x1,y1,x2,y2);

rectangle(xmin,ymin,xmax,ymax);

getch( );

OUTPUT:



**CONCLUSIONS:** A line is clipped using the algorithm

**SIGNATURE:**

**OPEN ENDED EXPERIMENT**

**DATE:**

**OBJECTIVE:** WAP to draw hyperbola

**SOFTWARE USED:** Turbo C3

**SOURCE CODE:**

#include<graphics.h>

#include<conio.h>

#include<stdio.h>

#include<dos.h>

#include<stdlib.h>

void main()

{

int gd=DETECT, gm;

initgraph(&gd,&gm,””);

float x,y;

float xc,yc;

float d,fx,fy,b,a;

d=b\*b\*(a+0.5)\*(a+0.5)-a\*a-a\*a\*b\*b;

printf(“enter centre (xc,yc)\n”);

scanf(“%f %f”,&xc,&yc);

printf(“enter a & b”);

scanf(“%f %f”,&a,&b);

x=a;

y=0;

fx=2\*b\*b\*a;

fy=0;

while(abs(fy)<=fx)

{ if(d>=0)

{

d=d-a\*a\*(2\*y+3);

}

else

{ d=d-a\*a\*(2\*y+3)+b\*b\*(2\*x+2);

x++;

fx=fx+2\*b\*b; }

y++;

fy=fy+2\*a\*a;

putpixel(x+320+xc,240-y-yc,GREEN);

putpixel(x+320+xc,240+y-yc,GREEN);

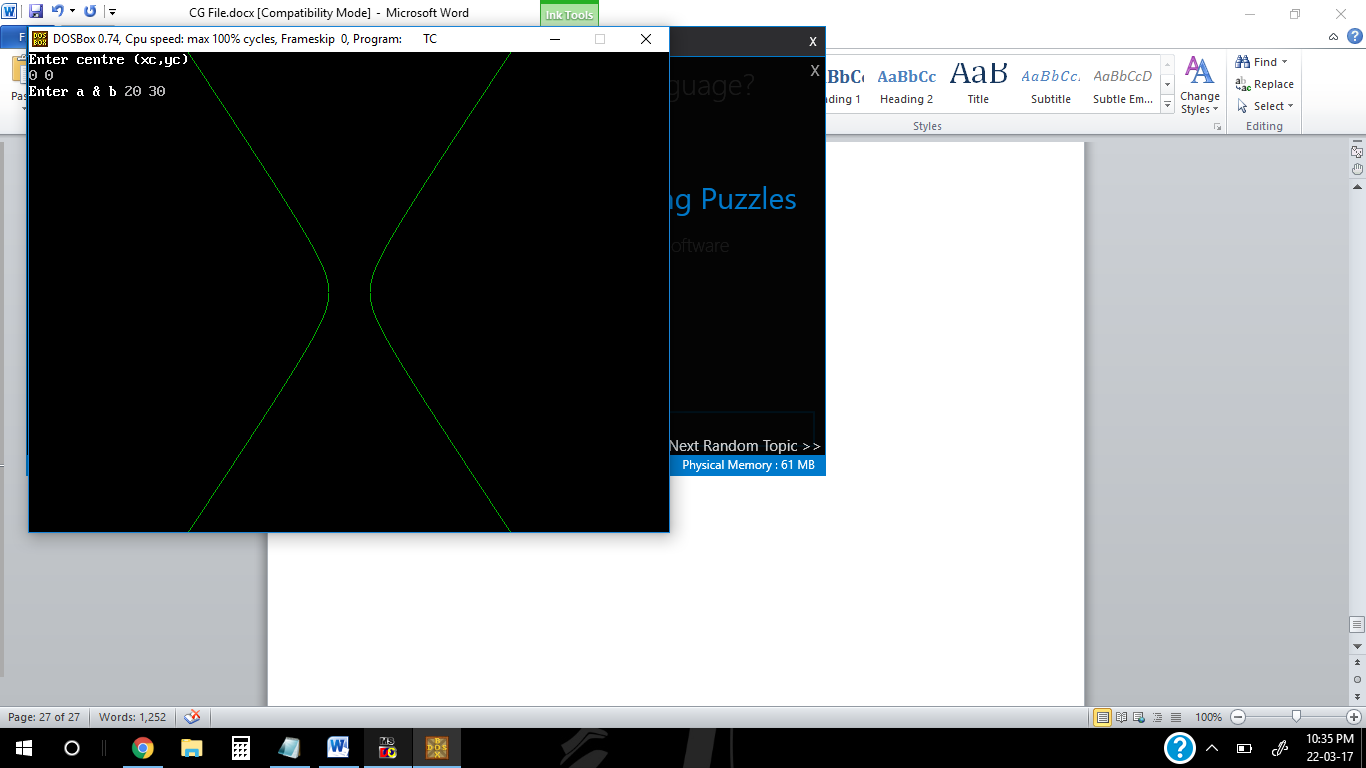
putpixel(-x+320+xc,240-y-yc,GREEN);

putpixel(-x+320+xc,240+y-yc,GREEN);}

getch();

}

**OUTPUT:**



**CONCLUSIONS:** A hyperbola is drawn on screen.

**SIGNATURE:**