**CG PRACTICAL 1 to 19**

**Prac\_1**

**Code :-**

**//program to drow a line ,triangle ,rectangle and home using graphics primitives.**

#include<graphics.h>

#include<conio.h>

void main()

{

int gd=DETECT,gm;

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

outtextxy(50,20,"Drawing a line...");

line(100,100,200,200);

getch();

cleardevice();

outtextxy(50,20,"Drawing a triangle...");

line(300,100,200,200);

line(300,100,400,200);

line(200,200,400,200);

getch();

cleardevice();

outtextxy(50,20,"Drawing a Rectangle...");

rectangle(125,115,215,165);

getch();

cleardevice();

outtextxy(50,20,"Drawing a Star...");

line(150,100,100,200);

line(100,200,200,200);

line(200,200,150,100);

line(100,125,200,125);

line(100,125,150,225);

line(150,225,200,125);

getch();

cleardevice();

outtextxy(50,20,"Drawing a House...");

setcolor(WHITE);

rectangle(150,180,250,300);

rectangle(250,180,420,300);

rectangle(180,250,220,300);

line(200,100,150,180);

line(200,100,250,180);

line(200,100,370,100);

line(370,100,420,180);

setfillstyle(SOLID\_FILL,BROWN);

floodfill(152,182,WHITE);

floodfill(252,182,WHITE);

setfillstyle(SLASH\_FILL,BLUE);

floodfill(182,252,WHITE);

setfillstyle(HATCH\_FILL,GREEN);

floodfill(200, 105,WHITE);

floodfill(210, 105,WHITE);

getch();

closegraph();

}

**Prac\_2**

**Code :-**

**//program to draw an arc, circle an ellipse and a smilley face.**

#include<conio.h>

#include<graphics.h>

#include<stdio.h>

void main()

{

int gd=DETECT,gm;

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

cleardevice();

outtextxy(50,20,"Drawing an arc...");

arc(120,200,180,0,30);

getch();

cleardevice();

outtextxy(50,20,"Drawing an circle...");

circle(120,270,30);

getch();

cleardevice();

outtextxy(50,20,"Drawing an ellipse...");

ellipse(120,350,0,360,30,20);

getch();

cleardevice();

int x =getmaxx()/2;

int y =getmaxy()/2;

outtextxy(x-100,20,"Drawing an circle...");

circle(x,y,80);

getch();

cleardevice();

outtextxy(50,20,"Drawing an ellipse...");

setcolor(YELLOW);

circle(300,100,40);

setfillstyle(SOLID\_FILL,YELLOW);

floodfill(300,100,YELLOW);

//Set color bacground is black

setcolor(BLACK);

setfillstyle(SOLID\_FILL,BLACK);

//create eye

fillellipse(310,82,2,6);

fillellipse(290,85,2,6);

//craete mouth

ellipse(300,100,205,335,20,9);

ellipse(300,100,205,335,20,10);

ellipse(300,100,205,335,20,11);

getch();

closegraph();

}

**Prac\_3**

**Code :-**

**//Wap to implement DDA line drawing algorithem.**

#include<stdio.h> //used for variable declaration etc...

#include<graphics.h> //used for putpixel function

#include<math.h> //used for getch and clrscr function

#include<iostream.h>

#include<conio.h>

#include<dos.h>

void main()

{

float x,y,dx,dy;

int i,x1,y1,x2,y2,length,sdx,sdy,gd,gm;

clrscr();

cout<<"\n Enter values of x1 & y1:\t";

cin>>x1>>y1;

cout<<"\n Enter the value of x2 & y2: \t";

cin>>x2>>y2;

gd=DETECT;

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

cleardevice();

line(120,240,520,240);

line(320,40,320,440);

line(320+x1,240-y1,320+x2,240-y2);

if(abs(x2-x1)>=abs(y2-y1))

{

length=abs(x2-x1);

}

else

{

length=abs(y2-y1);

}

dx=(x2-x1)/length;

dy=(y2-y1)/length;

if(dx>=0)

sdx=1;

else

sdx=-1;

if(dy>=0)

sdy=1;

else

sdy=-1;

x=x1+0.5\*sdx;

y=y1+0.5\*sdy;

i=1;

while(i<=length)

{

putpixel(x+320,240-y,5);

x=x+dx;

y=y+dy;

i=i+1;

delay(100);

}

getch();

closegraph();

}

**Prac\_4**

**Code :-**

// WAP to perform gene

#include<stdio.h>

#include<graphics.h>

#include<conio.h>

#include<math.h>

#include<iostream.h>

#include<dos.h>

addline(int x1,int y1,int x2,int y2)

{

float x=x1,y=y1,dx,dy;

int length,i;

putpixel(x1,y1,WHITE);

if(abs(x2-x1)>=abs(y2-y1))

length=abs(x2-x1);

else

length=abs(y2-y1);

dx=(float)(x2-x1)/length;

dy=(float)(y2-y1)/length;

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

{

x=x+dx;

y=y+dy;

putpixel((int)x,(int)y,WHITE);

delay(10);

}

return(0);

}

void main()

{

int x1,x2,y1,y2;

clrscr();

int gd=DETECT,gm;

cout<<"\n Enter the value of x1 & y1:";

cin>>x1>>y1;

cout<<"\n Enter the value of x2 & y2:";

cin>>x2>>y2;

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

addline(x1,y1,x2,y2);

getch();

}

**Prac\_5**

**Code :-**

#include<stdio.h>

#include<iostream.h>

#include<conio.h>

#include<dos.h>

#include<graphics.h>

#include<math.h>

void vla(int x, int y, int x2, int y2)

{

int dx,dy,t;

dx=x2-x;

dy=y2-y;

t=2(dy)-(dx);

while(x<=x2)

{

if(t<0)

{

x=x+1;

y=y;

t=t+2\*(dy);

}

else

{

x=x+1;

y=y+1;

t=t+2\*(dy-dx);

}

putpixel(x,y,WHITE);

delay(100);

}

}

void main()

{

int x1,x2,y1,y2;

int gd=DETECT,gm;

printf("\n Enter the value of x1 & y1\n");

scanf("%d%d"&x1,&y1);

printf("\n Enter the value of x2 & y2\n");

scanf("%d%d"&x2&y2);

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

vla(x1,y1,x2,y2);

getch();

}

**Prac\_6**

**Code :-**

**//program to draw a line generalized (BLA)Bresemanance Line algorithm**

#include<stdio.h>

#include<math.h>

#include<iostream.h>

#include<conio.h>

#include<dos.h>

#include<graphics.h>

void main()

{

int sdx,sdy,dx,dy,x1,x2,y1,y2,swap,i,e,x,y,temp;

int gd=DETECT,gm;

printf("\n Enter the value of x1 & y1\n");

scanf("%d%d", &x1,&y1);

printf("\n Enter the value of x2 & y2\n");

scanf("%d%d",&x2,&y2);

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

cleardevice();

line(120,240,520,240);

line(320,40,320,440);

line(320+x1,240+y1,320+x2,240+y2);

x=x1;

y=y1;

dx=abs(x2-x1);

dy=abs(y2-y1);

if(x2-x1>=0)

sdx=1;

else

sdx=-1;

if(y2-y1>=0)

sdy=1;

else

sdy=-1;

if(dy>dx)

{

temp=dx;

dx=dy;

dy=temp;

swap=1;

}

else

swap=0;

e=2\*dy-dx;

for(i=1;i<=dx;i++)

{

putpixel(x+320,240-y,10);

while(e>=0)

{

if(swap==0)

y=y+sdy;

else

x=x+sdx;

e=e-2\*dx;

}

if(swap==0)

x=x+sdx;

else

y=y+sdy;

e=e+2\*dy;

delay(100);

}

closegraph();

getch();

}

**Prac\_7**

**Code :-**

**// WAP an algorithm,draw a flowchart to draw a circle.**

#include<graphics.h>

#include<stdlib.h>

#include<stdio.h>

#include<conio.h>

#include<math.h>

#include<dos.h>

void Eightwaysymmetricplot(int xc,int yc, int x, int y)

{

putpixel(x+xc,y+yc,RED);

putpixel(x+xc,y+yc,YELLOW);

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

putpixel(-x+xc,-y+yc,YELLOW);

putpixel(y+xc,x+yc,12);

putpixel(y+xc,-x+yc,14);

putpixel(-y+xc,-x+yc,15);

putpixel(-y+xc,x+yc,6);

}

void Bresenhamcircle(int xc,int yc,int r)

{

int x=0,y=r,d=3-(2\*r);

Eightwaysymmetricplot(xc,yc,x,y);

while(x<=y)

{

if(d<=0)

{

d=d+(4\*x)+6;

}

else

{

d=d+(4\*x)-(4\*y)+10;

y=y-1;

}

x=x+1;

Eightwaysymmetricplot(xc,yc,x,y);

}

delay(100);

}

void main()

{

int xc,yc,r,gdriver=DETECT,gmode;

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

printf("\n Enter the value of xc & yc");

scanf("%d%d",&xc,&yc);

printf("\n Enter the value of radius=");

scanf("%d",&r);

Bresenhamcircle(xc,yc,r);

getch();

closegraph();

}

**Prac\_8**

**Code :-**

**// WAP an algorithm and draw a flowchart to draw a circle using midpoint circle drawing.**

#include<iostream.h>

#include<graphics.h>

#include<conio.h>

#include<math.h>

#include<dos.h>

void main()

{

int x,y,r,gd,gm;

clrscr();

cout<<"\n Enter the radius=";

cin>>r;

gd=DETECT;

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

cleardevice();

line(120,240,520,240);

line(320,40,320,440);

x=r;

do

{

y=sqrt(r\*r-x\*x);

putpixel(320+x,240-y,10);

putpixel(320-x,240-y,10);

putpixel(320-x,240+y,10);

putpixel(320+x,240+y,10);

x=x-0.01;

delay(20);

}

while(x>=0);

closegraph();

getch();

}

**Prac\_9**

**Code :-**

**// WAP to draw an ellipse.**

#include<iostream.h>

#include<dos.h>

#include<conio.h>

#include<graphics.h>

void circle(int,int,int,int);

void circlepoints(int,int,int,int,int);

void circle(int x0,int y0,int r,int xc)

{

int x=0;

int y=r;

double d=1-r;

int gd,gm;

gd=DETECT;

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

circlepoints(x0,x,y0,y,xc);

while(y>x)

{

if(d<0)

{

d+=2\*x\*3;

}

else

{

d+=2\*(x-y)+5;

y--;

}

x++;

circlepoints(x0,x,y0,y,xc);

}

getch();

closegraph();

}

void circlepoints(int x0,int x,int y0,int y,int xc)

{

putpixel((x0+x),(y0+y),xc);

putpixel((x0+x),(y0-y),xc);

putpixel((x0-x),(y0-y),xc);

putpixel((x0-x),(y0+x),xc);

putpixel((x0+y),(y0+x),xc);

putpixel((x0+y),(y0-x),xc);

putpixel((x0-y),(y0+x),xc);

putpixel((x0-y),(y0-x),xc);

}

void main()

{

int m[16]={BLACK,BLUE,GREEN,CYAN,RED,YELLOW,WHITE,LIGHTCYAN,LIGHTGRAY

,BROWN,LIGHTBLUE,LIGHTGREEN,LIGHTRED,MAGENTA,LIGHTMAGENTA};

int x0,y0,r,xc;

cout<<"\n Enter the value of x0 & y0:";

cin>>x0>>y0;

cout<<"\n Enter the value of radius:";

cin>>r;

cout<<"\n Enter the color";

for(int i=0;i<16;i++)

{

cin>>m[i];

xc=m[i];

break;

}

circle(x0,y0,r,xc);

}

**Prac\_10**

**Code :-**

**//Program to draw a polygon.**

#include<iostream.h>

#include<conio.h>

#include<graphics.h>

#include<dos.h>

#include<process.h>

class POLY

{

int ax[10],ay[10],df\_pen\_x,df\_pen\_y,n,i;

public:

POLY()

{

}

void poly();

};

void POLY::poly()

{

cout<<"\n Enter no. of sides of polygon: ";

cin>>n;

if(n<3)

{

cout<<"\n Polygon size error.....";

exit(0);

}

else

{

cout<<"\n Enter value in ax:";

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

{

cin>>ax[i];

}

cout<<"\n Enter value in ay:";

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

{

cin>>ay[i];

}

int gd=DETECT,gm;

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

df\_pen\_x=ax[n];

df\_pen\_y=ay[n];

moveto(df\_pen\_x,df\_pen\_y);

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

{

lineto(ax[i],ay[i]);

}

}

}

void main()

{

POLY p;

p.poly();

getch();

}

**Prac\_11**

**Code :-**

**// WAP to implement seed-fill(8 connected boundry fill)algorithm**

#include<iostream.h>

#include<conio.h>

#include<graphics.h>

#include<dos.h>

void bfill(int x, int y, int fill, int bd)

{

int current;

current=getpixel(x,y);

if((current!=bd)&&(current!=fill))

{

putpixel(x,y,fill);

delay(10);

bfill(x+1,y,fill,bd);

bfill(x-1,y,fill,bd);

bfill(x,y+1,fill,bd);

bfill(x,y-1,fill,bd);

bfill(x+1,y+1,fill,bd);

bfill(x+1,y-1,fill,bd);

bfill(x-1,y+1,fill,bd);

bfill(x-1,y-1,fill,bd);

}

}

void main()

{

int gd,gm;

clrscr();

detectgraph(&gd,&gm);

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

rectangle(50,50,100,100);

bfill(52,55,2,15);

getch();

closegraph();

}

**Prac\_12**

**Code :-**

**// WAP to demonstrate seed-fill (4 connected boundry fill)algorithm**

#include<iostream.h>

#include<conio.h>

#include<graphics.h>

#include<dos.h>

void bfill(int x, int y, int fill, int bd)

{

int current;

current=getpixel(x,y);

if((current!=bd)&&(current!=fill))

{

putpixel(x,y,fill);

delay(19);

bfill(x+1,y,fill,bd);

bfill(x-1,y,fill,bd);

bfill(x,y+1,fill,bd);

bfill(x,y-1,fill,bd);

}

}

void main()

{

int gd,gm;

clrscr();

detectgraph(&gd,&gm);

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

rectangle(50,50,100,100);

bfill(52,55,2,15);

getch();

closegraph();

}

**Prac\_13**

**Code :-**

**//WAP to demonstrate seed-fill(4 connected flood fill)algorithm**

#include<iostream.h>

#include<conio.h>

#include<graphics.h>

#include<dos.h>

void ffill(int x, int y, int old, int fill)

{

int current;

current=getpixel(x,y);

if(current==old)

{

putpixel(x,y,fill);

delay(1);

ffill(x+1,y,old,fill);

ffill(x-1,y,old,fill);

ffill(x,y+1,old,fill);

ffill(x,y-1,old,fill);

}

}

void main()

{

int gd,gm;

int od=0;

clrscr();

detectgraph(&gd,&gm);

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

rectangle(100,100,150,150);

ffill(125,125,od,4);

getch();

closegraph();

}

**Prac\_14**

**Code :-**

**// WAP to demonstrate seed-fill(8 connected flood fill)algorithm**

#include<iostream.h>

#include<conio.h>

#include<graphics.h>

#include<dos.h>

void ffill(int x, int y, int old, int fill)

{

int current;

current=getpixel(x,y);

if(current==old)

{

putpixel(x,y,fill);

delay(1);

ffill(x+1,y,old,fill);

ffill(x-1,y,old,fill);

ffill(x,y+1,old,fill);

ffill(x,y-1,old,fill);

ffill(x+1,y+1,old,fill);

ffill(x+1,y-1,old,fill);

ffill(x-1,y+1,old,fill);

ffill(x-1,y-1,old,fill);

}

}

void main()

{

int gd,gm;

int od=0;

clrscr();

detectgraph(&gd,&gm);

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

rectangle(100,100,150,150);

ffill(125,125,od,4);

getch();

closegraph();

}

**Prac\_15**

**Code :-**

**// WAP to program to implement scan line fill algorithm**.

#include<iostream.h>

#include<conio.h>

#include<graphics.h>

#include<stdlib.h>

#include<dos.h>

class point

{

public:

int x,y;

};

class poly

{

private:

point p[20];

int inter[20],x,y;

int v,xmin,ymin,xmax,ymax;

public:

int c;

void read();

void calcs();

void display();

void ints(float);

void sort(int);

};

void poly::read()

{

int i;

cout<<"\n\t SCAN FILL ALGORITHM";

cout<<"\n Enter the n. of vertices of polygon:";

cin>>v;

if(v>2)

{

for(i=0;i<v;i++) //ACCEPT THE VERTICES

{

cout<<"\n Enter the coordinate :" ;//no. -"<<i+1"<<":";

//cout<<"\n\t x"<<(i+1)<<"=";

cin>>p[i].x;

//cout<<"\n\t y"<<(i+1)<<"=";

cin>>p[i].y;

}

p[i].x=p[0].x;

p[i].y=p[0].y;

xmin=xmax=p[0].x;

ymin=ymax=p[0].y;

}

else

{

cout<<"\n Enter valid no.of vertices:";

exit(0);

}

}

void poly::calcs()

{

for(int i=0;i<v;i++)

{

if(xmin>p[i].x)

xmin=p[i].x;

if(xmax<p[i].x)

xmax=p[i].x;

if(ymin>p[i].y)

ymin=p[i].y;

if(ymax<p[i].y)

ymax=p[i].y;

}

}

//display function

void poly::display()

{

int ch1;

float s,s2;

s=ymin+0.01;

delay(100);

cleardevice();

while(s<=ymax)

{

ints(s);

sort(s);

s++;

}

}

void poly::ints(float z)//DEFINE FUNCTION

{

int x1,x2,y1,y2,temp;

c=0;

for(int i=1;i<v;i++)

{

x1=p[i].x;

y1=p[i].y;

x2=p[i+1].x;

y2=p[i+1].y;

if(y2<y1)

{

temp=x1;

x1=x2;

x2=temp;

temp=y1;

y1=y2;

y2=temp;

}

if(z<=y2&&z>=y1)

{

if((y1-y2)==0)

x=x1;

else

{

x=((x2-x1)\*(z-y1)/(y2-y1));

x=x+x1;

}

if(x<=xmax && x>=xmin)

inter[c++]=x;

}

}

}

void poly::sort(int z)//sort function

{

int temp,j,i;

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

{

line(p[i].x,p[i].y,p[i+1].x,p[i+1].y);//used to make hollow outline

}

delay(100);

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

{

delay(100);

line(inter[i],z,inter[i+1],z);//used to fill the polygon....

}

}

void main()

{

int cl;

int gd=DETECT,gm;

initgraph(&gd,&gm,"c:/tc/bgi");

cleardevice();

poly x;

x.read();

x.calcs();

cout<<"\n\t Enter the colour u want:(0-15)->";//selecting colour

cin>>cl;

setcolor(cl);

x.display();

closegraph();

getch();

}

**Prac\_16**

**Code :-**

**// WAP to implement of translate transformation.**

#include<iostream.h>

#include<conio.h>

#include<graphics.h>

#include<dos.h>

#include<process.h>

class POLY

{

int ax[10],ay[10],df\_pen\_x,df\_pen\_y,n,i,tx,ty ;

public:

POLY()

{

}

void poly();

void translate();

};

void POLY::poly()

{

cout<<"\n Enter no. of sides of polygon: ";

cin>>n;

if(n<3)

{

cout<<"\n Polygon size error.....";

exit(0);

}

else

{

cout<<"\n Enter value in ax:";

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

{

cin>>ax[i];

}

cout<<"\n Enter value in ay:";

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

{

cin>>ay[i];

}

int gd=DETECT,gm;

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

df\_pen\_x=ax[n];

df\_pen\_y=ay[n];

moveto(df\_pen\_x,df\_pen\_y);

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

{

lineto(ax[i],ay[i]);

}

}

}

void POLY::translate()

{

cout<<"\n Enter value of tx=";

cin>>tx;

cout<<"\n Enter value of ty=";

cin>>ty;

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

{

ax[i]=ax[i]+tx;

ay[i]=ay[i]+ty;

}

df\_pen\_x=ax[n];

df\_pen\_y=ay[n];

moveto(df\_pen\_x,df\_pen\_y);

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

{

lineto(ax[i],ay[i]);

}

}

void main()

{

POLY p;

p.poly();

getch();

p.translate();

getch();

}

**Prac\_17**

**Code :-**

//WAP to implement 2D Scalling Transformation.

#include<iostream.h>

#include<conio.h>

#include<graphics.h>

#include<dos.h>

#include<process.h>

#include<math.h>

class POLY

{

int ax[10],ay[10],df\_pen\_x,df\_pen\_y,n,i,sx,sy;

float xnew[10],ynew[10];

public:

POLY()

{

sx=1;

sy=1;

}

void poly();

void scaling();

};

void POLY::poly()

{

cout<<"\n Enter no. of sides of polygon: ";

cin>>n;

if(n<3)

{

cout<<"\n Polygon size error.....";

exit(0);

}

else

{

cout<<"\n Enter value in ax:";

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

{

cin>>ax[i];

}

cout<<"\n Enter value in ay:";

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

{

cin>>ay[i];

}

int gd=DETECT,gm;

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

df\_pen\_x=ax[n];

df\_pen\_y=ay[n];

moveto(df\_pen\_x,df\_pen\_y);

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

{

lineto(ax[i],ay[i]);

}

}

}

void POLY::scaling()

{

cout<<"\n Enter value of sx=";

cin>>sx;

cout<<"\n Enter value of sy=";

cin>>sy;

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

{

xnew[i]=ax[i]\*sx;

ynew[i]=ay[i]\*sy;

}

df\_pen\_x=xnew[n];

df\_pen\_y=ynew[n];

moveto(df\_pen\_x,df\_pen\_y);

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

{

lineto(xnew[i],ynew[i]);

}

}

void main()

{

POLY p;

p.poly();

getch();

p.scaling();

getch();

}

**Prac\_18**

**Code :-**

//WAP to implement 2D Rotation transformation

#include<iostream.h>

#include<conio.h>

#include<graphics.h>

#include<dos.h>

#include<process.h>

#include<math.h>

class POLY

{

int ax[10],ay[10],df\_pen\_x,df\_pen\_y,n,i;

float xnew[10],ynew[10],xsum,ysum,xc,yc,angle;

public:

POLY()

{

xsum=0;

ysum=0;

}

void poly();

void rotate();

};

void POLY::poly()

{

cout<<"\n Enter no. of sides of polygon: ";

cin>>n;

if(n<3)

{

cout<<"\n Polygon size error.....";

exit(0);

}

else

{

cout<<"\n Enter value in ax:";

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

{

cin>>ax[i];

}

cout<<"\n Enter value in ay:";

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

{

cin>>ay[i];

}

int gd=DETECT,gm;

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

df\_pen\_x=ax[n];

df\_pen\_y=ay[n];

moveto(df\_pen\_x,df\_pen\_y);

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

{

lineto(ax[i],ay[i]);

}

}

}

void POLY::rotate()

{

cout<<"\n Enter the angle=";

cin>>angle;

angle=(3.14\*angle)/180;

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

{

xsum=xsum+ax[i];

ysum=ysum+ay[i];

}

xc=xsum/n;

yc=ysum/n;

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

{

xnew[i]=ax[i]\*cos(angle)-ay[i]\*sin(angle)-xc\*cos(angle)+yc\*sin(angle)+xc;

ynew[i]=ax[i]\*sin(angle)+ay[i]\*cos(angle)-xc\*sin(angle)-yc\*cos(angle)+yc;

}

setcolor(5);

df\_pen\_x=xnew[n];

df\_pen\_y=ynew[n];

moveto(df\_pen\_x,df\_pen\_y);

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

{

lineto(xnew[i],ynew[i]);

}

}

void main()

{

POLY p;

p.poly();

getch();

p.rotate();

getch();

}

**Prac\_19**

**Code :-**

//Program to implement 2D shearing Transformation......

#include<iostream.h>

#include<conio.h>

#include<math.h>

#include<process.h>

#include<dos.h>

#include<graphics.h>

class POLY

{

int ax[10],ay[10],DE\_PEN\_X,DF\_PEN\_Y,n,i,sx,sy;

float xnew[10],ynew[10];

public:

POLY()

{

sx=1;

sy=1;

}

void poly();

void xshearing();

void yshearing();

};

void POLY::poly()

{

cout<<"Enter no. of side of polygon : ";

cin>>n;

if(n<3)

{

cout<<"Polygon size error........";

exit(0);

}

else

{

cout<<"Enter the value in ax: ";

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

{

cin>>ax[i];

}

cout<<"Enter the value in ay: ";

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

{

cin>>ay[i];

}

int gd=DETECT,gm;

initgraph(&gd,&gm,"c:/tc/bgi");

DE\_PEN\_X=ax[n];

DF\_PEN\_Y =ay[n];

moveto(DE\_PEN\_X,DF\_PEN\_Y);

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

{

lineto(ax[i],ay[i]);

}

}

}

void POLY::xshearing()

{

cout<<"\n\n\t\t Enter the X-shear factor(Sx) : ";

cin>>sx;

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

{

xnew[i]=ax[i]+sx\*ay[i];

ynew[i]=ay[i];

}

DE\_PEN\_X=xnew[n];

DF\_PEN\_Y=ynew[n];

moveto(DE\_PEN\_X,DF\_PEN\_Y);

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

{

lineto(xnew[i],ynew[i]);

}

}

void POLY::yshearing()

{

cout<<"\n\n\n\t\t Enter the Y-shear factor(Sy) : ";

cin>>sy;

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

{

xnew[i]=ax[i];

ynew[i]=sy\*ax[i]+ay[i];

}

DE\_PEN\_X=xnew[n];

DF\_PEN\_Y=ynew[n];

moveto(DE\_PEN\_X,DF\_PEN\_Y);

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

{

lineto(xnew[i],ynew[i]);

}

}

void main()

{

POLY p;

p.poly();

getch();

p.xshearing();

getch();

p.yshearing();

getch();

}