1. **Basic Primitives:**
   1. **Line:**

Source Code:

#include <graphics.h>

#include <conio.h>

Int main()

{

int gd = DETECT, gm;

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

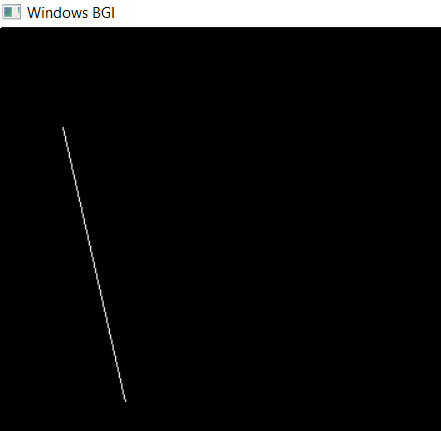
line(50,80,100,300);

getch();

closegraph();

return 0;

}

 Output:

* 1. **Circle:**

Source Code:

//C Implementation for Drawing Circle

#include <graphics.h>

int main()

{

int gd = DETECT, gm;

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

circle(150, 150, 100);

getch();

closegraph();

return 0;

}

Output:



* 1. **Ellipse**

Source Code:

// C Implementation for drawing ellipse

#include <graphics.h>

int main()

{

int gd = DETECT, gm;

int x = 250, y = 200;

int start\_angle = 0;

int end\_angle = 360;

int x\_rad = 100;

int y\_rad = 50;

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

ellipse(x, y, start\_angle,

end\_angle, x\_rad, y\_rad);

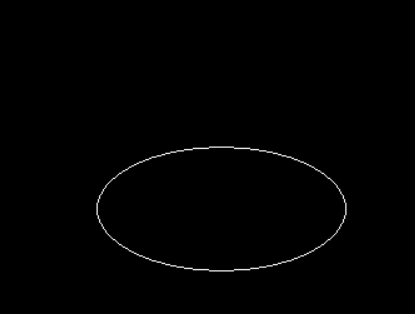
getch();

closegraph();

return 0;

}

Output:



* 1. **Rectangle**

Source Code:

// C program to draw a rectangle

#include <graphics.h>

int main()

{

int gd = DETECT, gm;

// location of left, top, right, bottom

int left = 150, top = 150;

int right = 450, bottom = 450;

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

// rectangle function

rectangle(left, top, right, bottom);

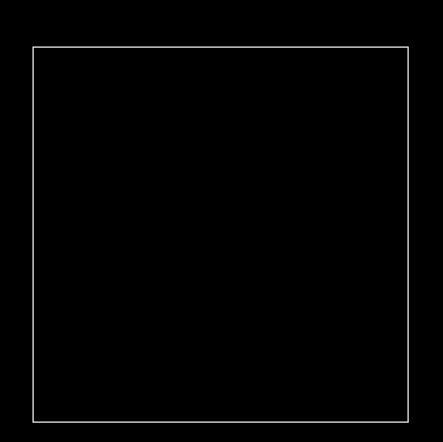
getch();

closegraph();

return 0;

}

Output:



* 1. **Polygon**

Source Code:

// C program to draw a Polygon

#include <graphics.h>

#include <conio.h>

int main()

{

int gd = DETECT, gm;

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

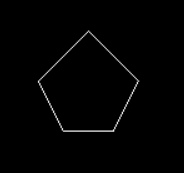
int pentagon[12] = {340,150,320,110,360,70,400,110,380,150,340,150};

drawpoly(6,pentagon);

getch();

}

Output:



**2. Digital Differential Analyzer (DDA):**

Source Code:

#include<stdio.h>

#include<graphics.h>

#include<math.h>

float round(float a);

int main()

{

int gd=DETECT,gm;

// gd=graphics driver (detects best graphics driver and assigns it as default, gm=graphics mode.

int x1,y1,x2,y2,steps,k;

float xincr,yincr,x,y,dx,dy;

printf("enter x1,y1");

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

printf("enter x2,y2");

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

initgraph(&gd,&gm,"c:\\turboc3\\BGI");//initializes the graph

dx=x2-x1;

dy=y2-y1;

if(abs(dx)>abs(dy))

steps=abs(dx);

else

steps=abs(dy);

xincr=dx/steps;

yincr=dy/steps;

x=x1;

y=y1;

for(k=1;k<=steps;k++)

{

delay(100);//for seeing the line drawing process slowly.

x+=xincr;

y+=yincr;

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

}

outtextxy(200,20,"DDA"); // for printing text at desired screen location.

outtextxy(x1+5,y1-5,"(x1,y1)");

outtextxy(x2+5,y2+5,"(x2,y2)");

getch();

}

float round(float a)

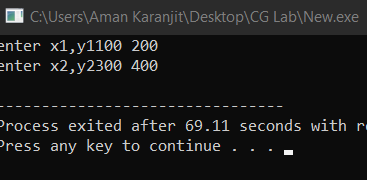
{

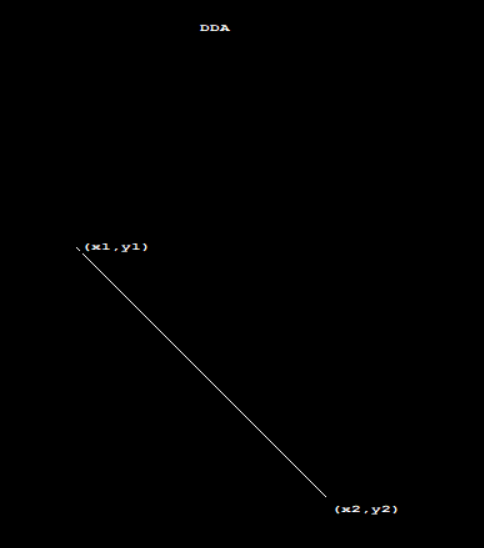
int b=a+0.5;

return b;

}

Output:





**3. Bresenham’s Line Drawing Algorithm (BLA):**

Source Code:

#include<stdio.h>

#include<conio.h>

#include<graphics.h>

int main()

{

int x,y,x1,y1,x2,y2,p,dx,dy;

int gd=DETECT,gm;

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

printf("\nEnter the x-coordinate of the first point ::");

scanf("%d",&x1);

printf("\nEnter the y-coordinate of the first point ::");

scanf("%d",&y1);

printf("\nEnter the x-coordinate of the second point ::");

scanf("%d",&x2);

printf("\nEnter the y-coordinate of the second point ::");

scanf("%d",&y2);

x=x1;

y=y1;

dx=x2-x1;

dy=y2-y1;

putpixel(x,y,2);

p=(2\*dy-dx);

while(x<=x2)

{

if(p<0)

{

x=x+1;

p=p+2\*dy;

}

else

{

x=x+1;

y=y+1;

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

}

putpixel(x,y,7);

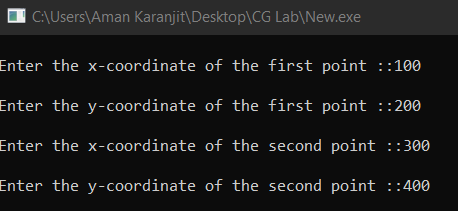
}

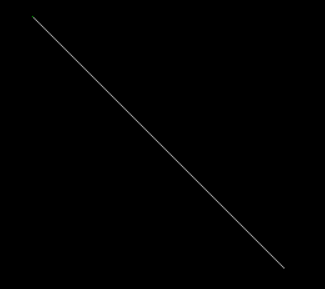
getch();

closegraph();

}

Output:





**4. Mid-Point Circle Drawing Algorithm:**

Source Code:

#include<stdio.h>

#include<conio.h>

#include<graphics.h>

int draw\_circle(int,int,int);

int symmetry(int,int,int,int);

int main()

{

int xc,yc,R;

int gd=DETECT,gm;

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

printf("Enter the center of the circle:\n");

printf("Xc =");

scanf("%d",&xc);

printf("Yc =");

scanf("%d",&yc);

printf("Enter the radius of the circle :");

scanf("%d",&R);

draw\_circle(xc,yc,R);

getch();

closegraph();

}

int draw\_circle(int xc,int yc,int rad)

{

int x = 0;

int y = rad;

int p = 1-rad;

symmetry(x,y,xc,yc);

for(x=0;y>x;x++)

{

if(p<0)

p += 2\*x + 3;

else

{

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

y--;

}

symmetry(x,y,xc,yc);

delay(20);

}

}

int symmetry(int x,int y,int xc,int yc)

{

putpixel(xc+x,yc-y,WHITE); //For pixel (x,y)

delay(20);

putpixel(xc+y,yc-x, WHITE); //For pixel (y,x)

delay(20);

putpixel(xc+y,yc+x, WHITE); //For pixel (y,-x)

delay(20);

putpixel(xc+x,yc+y, WHITE); //For pixel (x,-y)

delay(20);

putpixel(xc-x,yc+y, WHITE); //For pixel (-x,-y)

delay(20);

putpixel(xc-y,yc+x, WHITE); //For pixel (-y,-x)

delay(20);

putpixel(xc-y,yc-x, WHITE); //For pixel (-y,x)

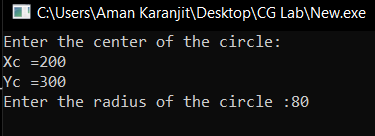
delay(20);

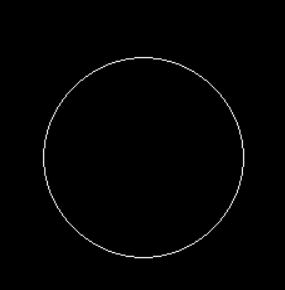
putpixel(xc-x,yc-y, WHITE); //For pixel (-x,y)

delay(20);

}

Output:





**5. Translation:**

Source Code:

#include<stdio.h>

#include<conio.h>

#include<graphics.h>

#include<math.h>

int main()

{

int gd=DETECT,gm;

int x1,y1,x2,y2,tx,ty,x3,y3,x4,y4;

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

printf("Enter the starting point of line segment:");

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

printf("Enter the ending point of line segment:");

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

printf("Enter translation distances tx,ty:\n");

scanf("%d%d",&tx,&ty);

setcolor(5);

line(x1,y1,x2,y2);

outtextxy(x2+2,y2+2,"Original line");

x3=x1+tx;

y3=y1+ty;

x4=x2+tx;

y4=y2+ty;

setcolor(7);

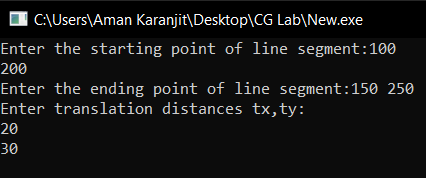
line(x3,y3,x4,y4);

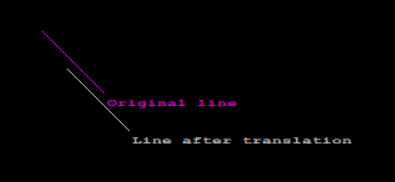
outtextxy(x4+2,y4+2,"Line after translation");

getch();

}

Output:





**6. Rotation:**

Source Code:

#include<graphics.h>

#include<stdio.h>

#include<conio.h>

#include<math.h>

int main()

{

int gd=DETECT,gm;

int pivot\_x,pivot\_y,x,y;

double degree,radian;

int rotated\_point\_x,rotated\_point\_y;

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

cleardevice();

printf("\t\t\*\*\*\*\*\*\*\*\*\*\* ROTATION \*\*\*\*\*\*\*\*\*\*\* \n");

printf("\n Enter an initial coordinates of the line = ");

scanf("%d %d",&pivot\_x,&pivot\_y);

printf("\n Enter a final coordinates of the line = ");

scanf("%d %d",&x,&y);

line(pivot\_x,pivot\_y,x,y);

printf("\n\n Now, Enter a degree = ");

scanf("%lf",&degree);

radian=degree\*0.01745;

rotated\_point\_x=(int)(pivot\_x +((x-pivot\_x)\*cos(radian)-(y-pivot\_y)\*sin(radian)));

rotated\_point\_y=(int)(pivot\_y +((x-pivot\_x)\*sin(radian)+(y-pivot\_y)\*cos(radian)));

setcolor(RED);

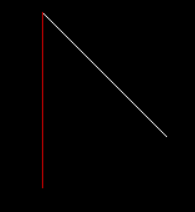
line(pivot\_x,pivot\_y,rotated\_point\_x,rotated\_point\_y);

getch();

closegraph();

}

Output:



**7. Scaling:**

Source Code:

#include<stdio.h>

#include<graphics.h>

#include<math.h>

int graDriver=DETECT,graMode;

int n,xs[100],ys[100],i;

float sfx,sfy;

void DrawFn()

{

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

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

}

void scale()

{

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

{

xs[i]=xs[0]+(int)((float)(xs[i]-xs[0])\*sfx);

ys[i]=ys[0]+(int)((float)(ys[i]-ys[0])\*sfy);

}

}

int 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("Enter scale factors: (xs,ys) ");

scanf("%f%f",&sfx,&sfy);

initgraph(&graDriver,&graMode,"C:\\TURBOC3\\BGI\\");

setcolor(RED);

DrawFn();//original

scale();//scaling

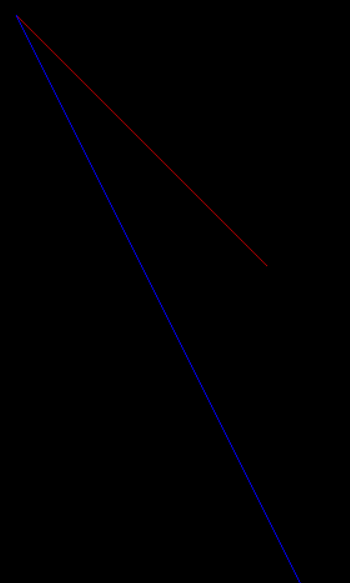
setcolor(BLUE);

DrawFn();

getch();

}

Output:



**8. Reflection:**

Source Code:

#include <conio.h>

#include <graphics.h>

#include <stdio.h>

int main()

{

int gm, gd = DETECT, ax, x1 = 100;

int x2 = 100, x3 = 200, y1 = 100;

int y2 = 200, y3 = 100;

// Add in your BGI folder path

// like below initgraph(&gd, &gm,

// "C:\\TURBOC3\\BGI");

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

cleardevice();

line(getmaxx() / 2, 0, getmaxx() / 2,

getmaxy());

line(0, getmaxy() / 2, getmaxx(),

getmaxy() / 2);

printf("Before Reflection Object"

" in 2nd Quadrant");

setcolor(14);

line(x1, y1, x2, y2);

line(x2, y2, x3, y3);

line(x3, y3, x1, y1);

printf("\nAfter Reflection");

setcolor(4);

line(getmaxx() - x1, getmaxy() - y1,

getmaxx() - x2, getmaxy() - y2);

line(getmaxx() - x2, getmaxy() - y2,

getmaxx() - x3, getmaxy() - y3);

line(getmaxx() - x3, getmaxy() - y3,

getmaxx() - x1, getmaxy() - y1);

setcolor(3);

line(getmaxx() - x1, y1,

getmaxx() - x2, y2);

line(getmaxx() - x2, y2,

getmaxx() - x3, y3);

line(getmaxx() - x3, y3,

getmaxx() - x1, y1);

setcolor(2);

line(x1, getmaxy() - y1, x2,

getmaxy() - y2);

line(x2, getmaxy() - y2, x3,

getmaxy() - y3);

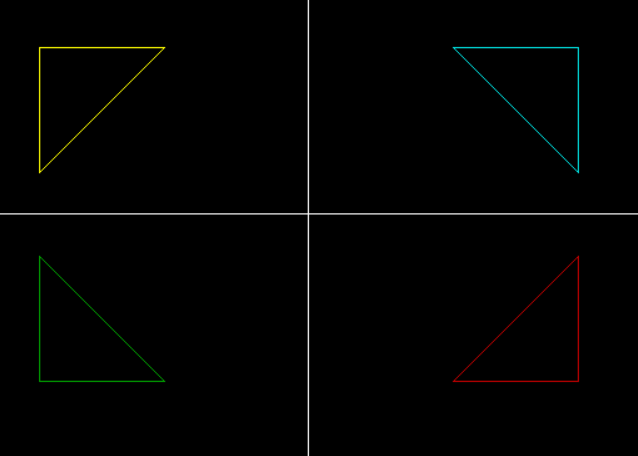
line(x3, getmaxy() - y3, x1,

getmaxy() - y1);

getch();

closegraph();

}

Output: 

**9. Rotation about fixed Point**

Source Code:

#include<stdio.h>

#include<conio.h>

#include<graphics.h>

#include<stdlib.h>

int main()

{

int gd=DETECT,gm,x,y,ch;

char p[10];

int i,t;

initgraph(&gd,&gm,"c:\\turboc3\\bgi");

setbkcolor(BLUE);

outtextxy(480,400,"x");

outtextxy(150,90,"y");

line(150,100,150,400);

line(150,400,470,400);

outtextxy(140,405,"0");

for(i=0,t=25;i<12;i++)

{

itoa(i+1,p,10);

outtextxy(150+t,410,p);

line(150+t,400,150+t,405);

itoa(i+1,p,10);

outtextxy(130,400-t,p);

line(150,400-t,145,400-t);

t+=25;

}

setcolor(5);

line(250,200,350,200);

line(250,200,300,300);

line(300,300,350,200);

x=(250+350+300)/3;

y=(200+200+300)/3;

putpixel(x,y,10);

setcolor(RED);

setlinestyle(2,0,2);

line(150,467,200,367);

line(100,367,150,467);

line(100,367,200,367);

getch();

setbkcolor(RED);

setcolor(YELLOW);

line(200,350,150-67,400);

line(200,350,200,450);

line(200,450,150-67,400);

setcolor(YELLOW);

getch();

setbkcolor(RED);

line(300-67,233,350,183);

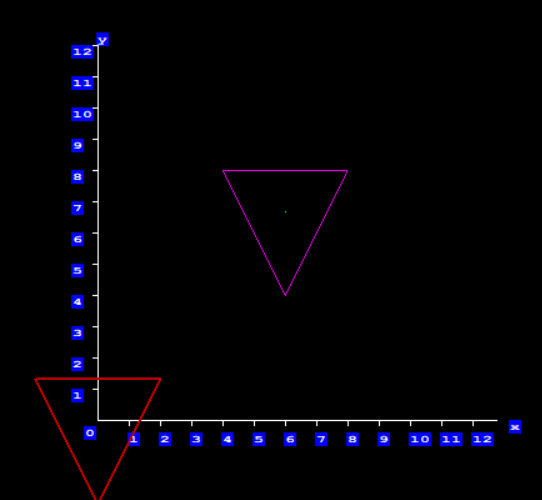
line(350,183,350,283);

line(350,283,300-67,233);

getch();

}

Output:



**10. Scaling about fixed point**

Source Code:

//program for fixed point translation & fixed point scaling

#include<stdio.h>

#include<conio.h>

#include<graphics.h>

#include<dos.h>

#include<stdlib.h>

int main()

{

int gd=DETECT,gm,x,y,ch;

char p[10];

int i,t;

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

setbkcolor(BLUE);

outtextxy(480,400,"x");

outtextxy(150,90,"y");

line(150,100,150,400);

line(150,400,470,400);

outtextxy(140,405,"0");

for(i=0,t=25;i<12;i++)

{

itoa(i+1,p,10);

outtextxy(150+t,410,p);

line(150+t,400,150+t,405);

itoa(i+1,p,10);

outtextxy(130,400-t,p);

line(150,400-t,145,400-t);

t+=25;

}

setcolor(5);

line(250,200,350,200);

line(250,200,300,300);

line(300,300,350,200);

x=(250+350+300)/3;

y=(200+200+300)/3;

putpixel(x,y,10);

printf("Enter your choice");

printf("\npress 1: fixed point Rotation");

printf("\npress 2: fixed point Scaling");

printf("\nchoice=");

scanf("%d",&ch);

switch(ch)

{

case 1: setcolor(RED);

setlinestyle(2,0,2);

line(150,467,200,367);

line(100,367,150,467);

line(100,367,200,367);

getch();

setbkcolor(RED);

setcolor(YELLOW);

line(200,350,150-67,400);

line(200,350,200,450);

line(200,450,150-67,400);

setcolor(YELLOW);

getch();

setbkcolor(RED);

line(300-67,233,350,183);

line(350,183,350,283);

line(350,283,300-67,233);

break;

case 2: setcolor(RED);

setlinestyle(2,0,2);

line(150,467,200,367);

line(100,367,150,467);

line(100,367,200,367);

getch();

setcolor(YELLOW);

line(150,447,180,377);

line(180,377,120,377);

line(120,377,150,447);

setbkcolor(RED);

getch();

setcolor(YELLOW);

line(270,210,330,210);

line(270,210,300,280);

line(300,280,330,210);

getch();

break;

default : exit(0);

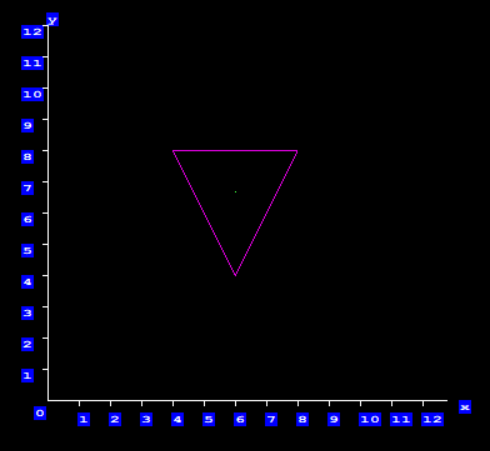
break;

}

getch();

}

Output:



**11. Scan Line Polygon Fill Algorithm**

Source Code:

#include <stdio.h>

#include <conio.h>

#include <graphics.h>

int main()

{

int n,i,j,k,gd,gm,dy,dx;

int x,y,temp;

int a[20][2],xi[20];

float slope[20];

printf("\n\n\tEnter the no. of edges of polygon : ");

scanf("%d",&n);

printf("\n\n\tEnter the cordinates of polygon :\n\n\n ");

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

{

printf("\tX%d Y%d : ",i,i);

scanf("%d %d",&a[i][0],&a[i][1]);

}

a[n][0]=a[0][0];

a[n][1]=a[0][1];

detectgraph(&gd,&gm);

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

/\*- draw polygon -\*/

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

{

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

}

getch();

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

{

dy=a[i+1][1]-a[i][1];

dx=a[i+1][0]-a[i][0];

if(dy==0) slope[i]=1.0;

if(dx==0) slope[i]=0.0;

if((dy!=0)&&(dx!=0)) /\*- calculate inverse slope -\*/

{

slope[i]=(float) dx/dy;

}

}

for(y=0;y< 480;y++)

{

k=0;

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

{

if( ((a[i][1]<=y)&&(a[i+1][1]>y))||

((a[i][1]>y)&&(a[i+1][1]<=y)))

{

xi[k]=(int)(a[i][0]+slope[i]\*(y-a[i][1]));

k++;

}

}

for(j=0;j<k-1;j++) /\*- Arrange x-intersections in order -\*/

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

{

if(xi[i]>xi[i+1])

{

temp=xi[i];

xi[i]=xi[i+1];

xi[i+1]=temp;

}

}

setcolor(3);

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

{

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

getch();

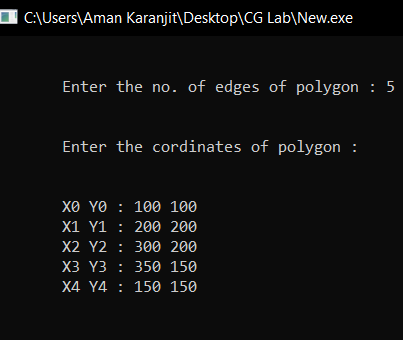
}

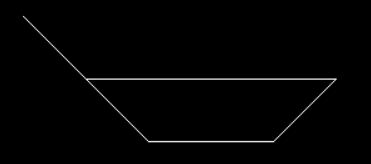
}

}55

11

Output:





**12. Shearing:**

Source Code:

#include<stdio.h>

#include<graphics.h>

#include<math.h>

int graDriver=DETECT,graMode;

int n,xs[100],ys[100],i;

float shearXfactor,shearYfactor;

void DrawFn()

{

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

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

}

void shearAlongX()

{

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

xs[i]=xs[i]+shearXfactor\*ys[i];

}

void shearAlongY()

{

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

ys[i]=ys[i]+shearYfactor\*xs[i];

}

int 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("Enter x shear factor:");

scanf("%f",&shearXfactor);

printf("Enter y shear factor:");

scanf("%f",&shearYfactor);

initgraph(&graDriver,&graMode,"C:\\TURBOC3\\BGI\\");

setcolor(RED);

DrawFn();//original

shearAlongX();

setcolor(BLUE);

DrawFn();//Xshear

shearAlongY();

setcolor(GREEN);

DrawFn();//Yshear

getch();

}

Output:

