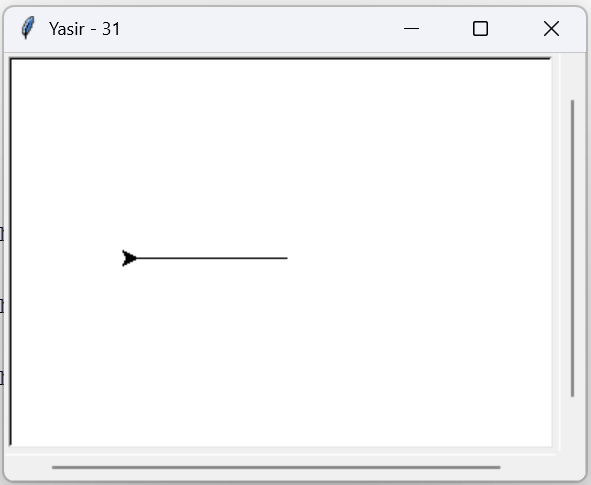
1a.

import turtle

turtle.title("Yasir - 31")

turtle.forward(100)

turtle.done()

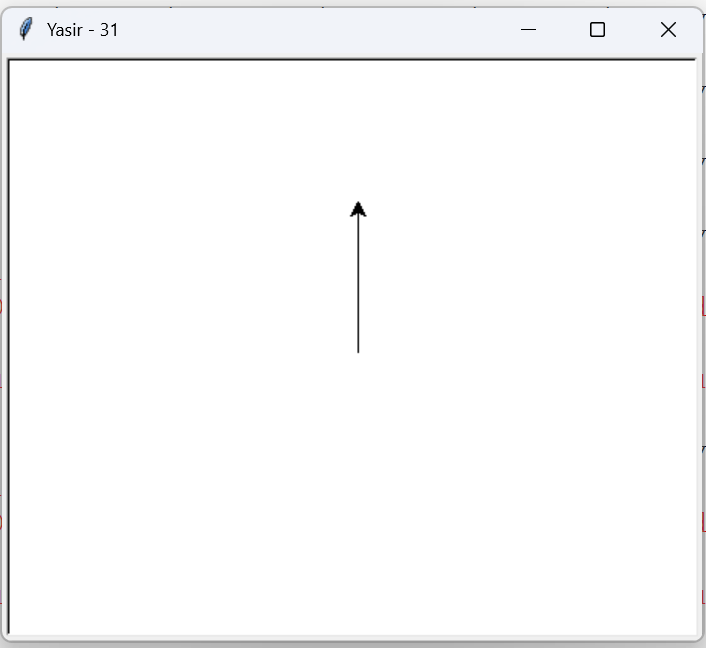
1b.

import turtle

turtle.title("Yasir - 31")

turtle.backward(100)

turtle.done()

1c.

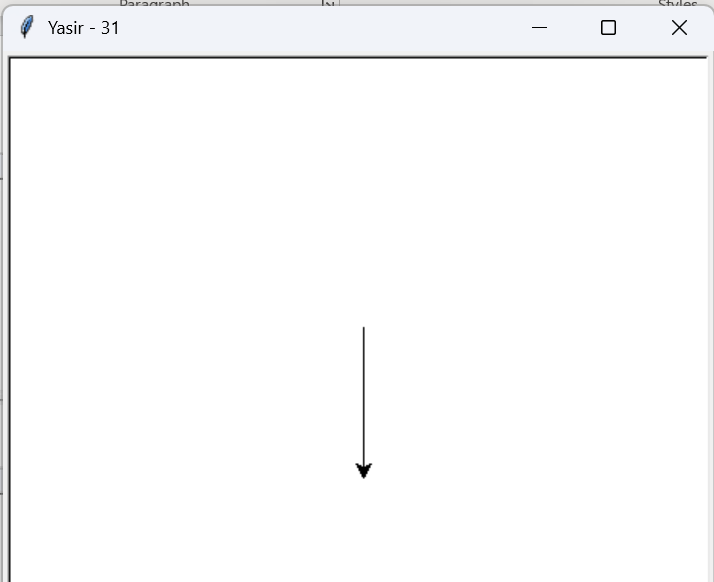
import turtle

turtle.title("Yasir - 31")

turtle.left(90)

turtle.forward(100)

turtle.done()

1d.

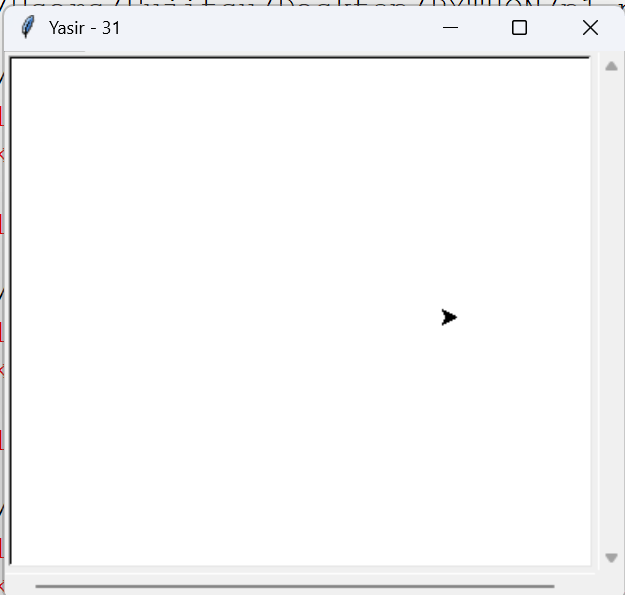
import turtle

turtle.title("Yasir - 31")

turtle.right(90)

turtle.forward(100)

turtle.done()



1e.

import turtle

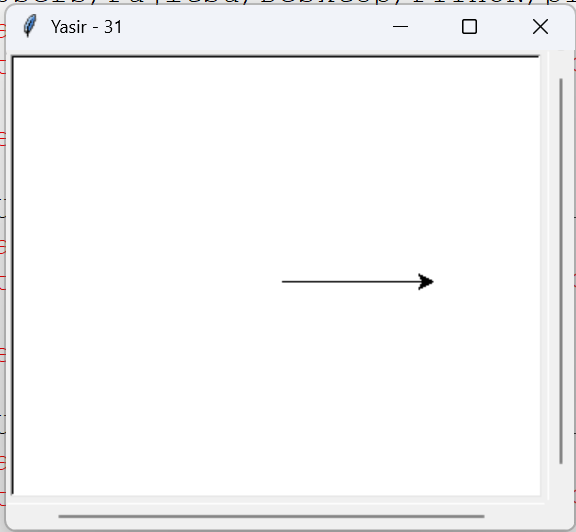
turtle.title("Yasir - 31")

turtle.penup()

turtle.forward(100)

turtle.done()

1f.

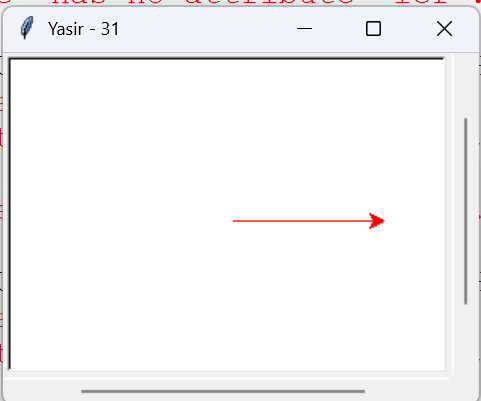
import turtle

turtle.title("Yasir - 31")

turtle.pendown()

turtle.forward(100)

turtle.done()



1g.

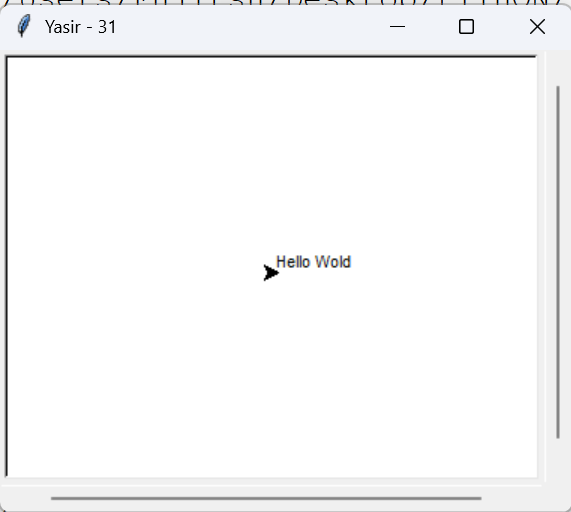
import turtle

turtle.title("Yasir - 31")

turtle.color("red")

turtle.forward(100)

turtle.done()



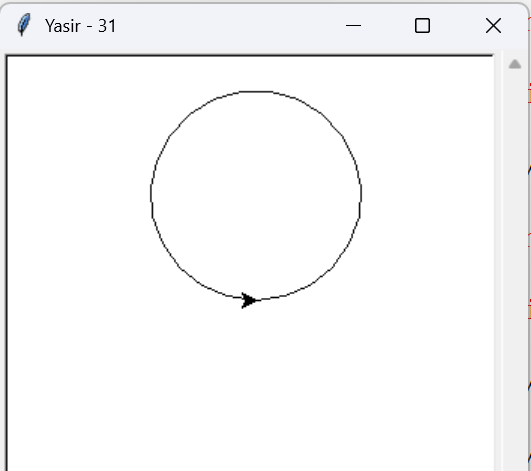
1h.

import turtle

turtle.title("Yasir - 31")

turtle.write("Hello Wold")

turtle.done()

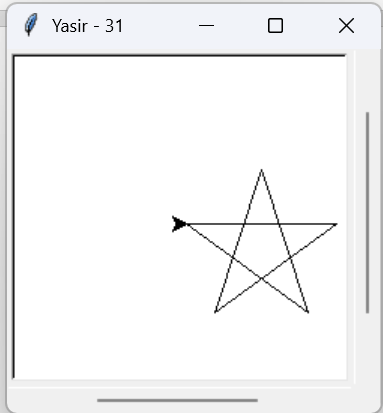
1i.

import turtle

turtle.title("Yasir - 31")

turtle.circle(70)

turtle.done()



1j.

import turtle

turtle.title("Yasir - 31")

turtle.speed(1)

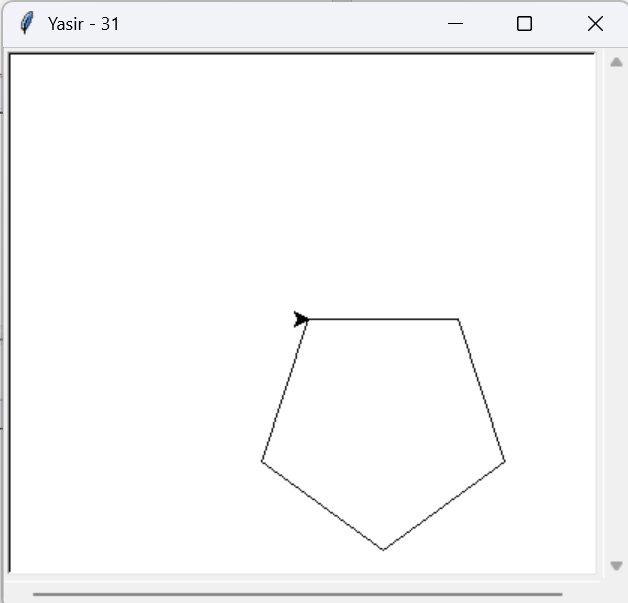
for \_ in range(5):

turtle.forward(100)

turtle.right(144)

turtle.done()

1k.

import turtle

turtle.title("Yasir - 31")

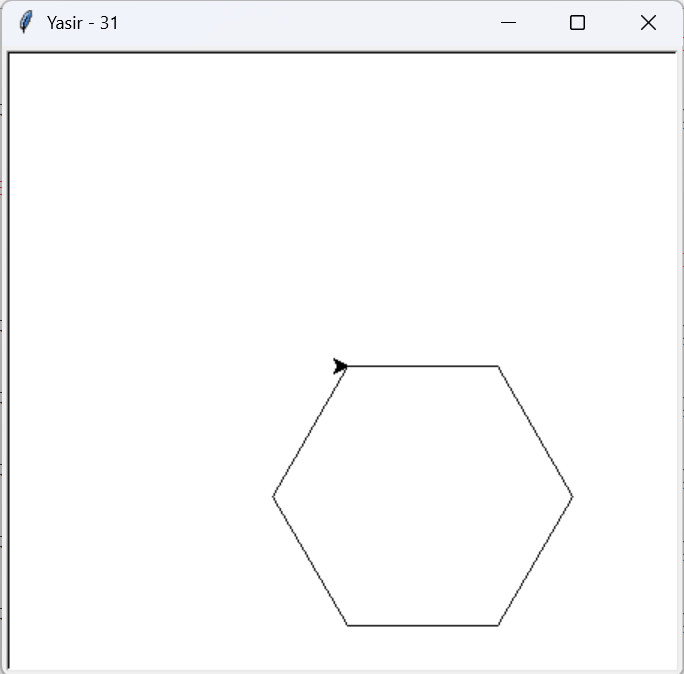
turtle.speed(1)

for \_ in range(5):

turtle.forward(100)

turtle.right(72)

turtle.done()



1l.

import turtle

turtle.title("Yasir - 31")

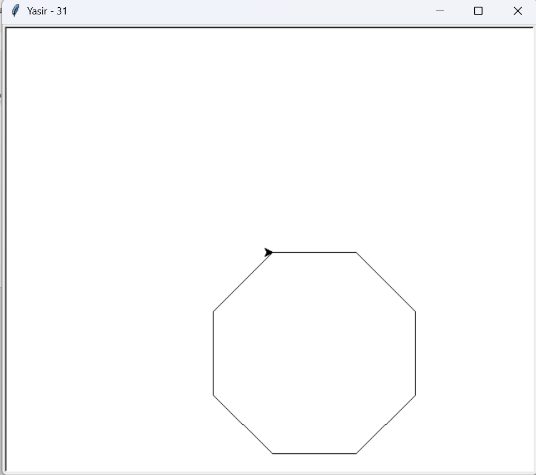
turtle.speed(1)

for \_ in range(6):

turtle.forward(100)

turtle.right(60)

turtle.done()

1m.

import turtle

turtle.title("Yasir - 31")

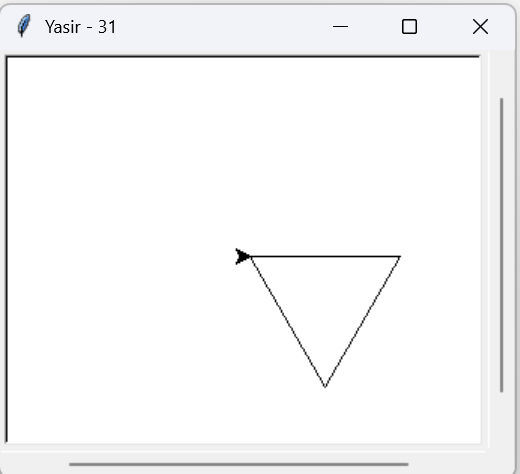
turtle.speed(1)

for \_ in range(8):

turtle.forward(100)

turtle.right(45)

turtle.done()

1n.

import turtle

turtle.title("Yasir - 31")

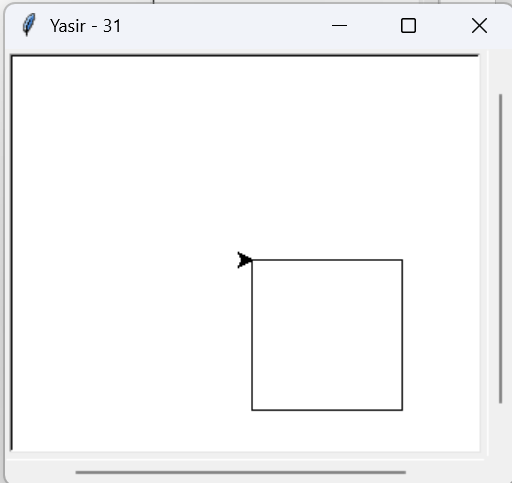
turtle.speed(1)

for \_ in range(3):

turtle.forward(100)

turtle.right(120)

turtle.done()



1o.

import turtle

turtle.title("Yasir - 31")

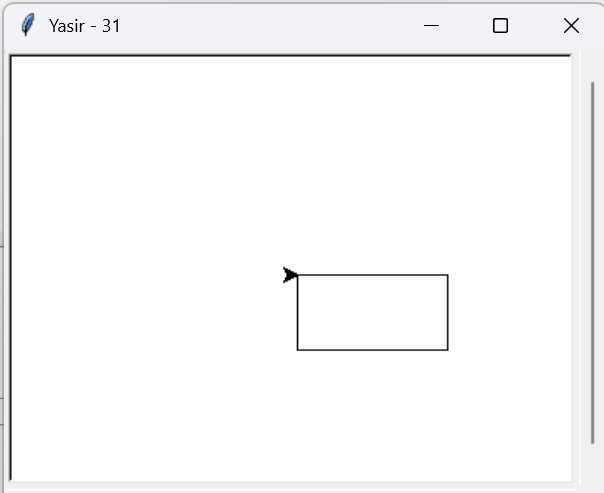
turtle.speed(1)

for \_ in range(4):

turtle.forward(100)

turtle.right(90)

turtle.done()

1p.

import turtle

turtle.title("Yasir - 31")

turtle.speed(1)

for \_ in range(2):

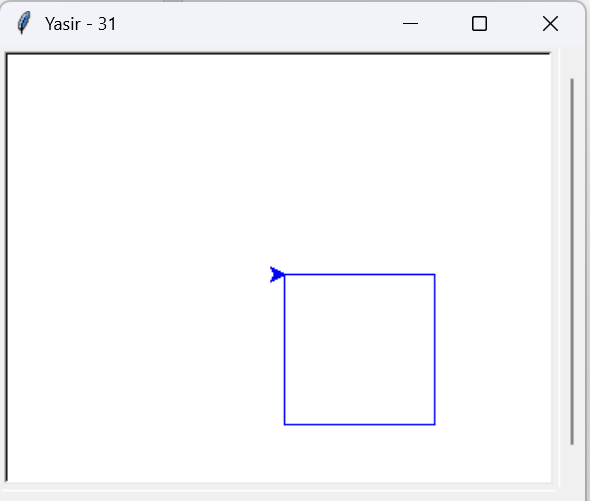
turtle.forward(100)

turtle.right(90)

turtle.forward(50)

turtle.right(90)

turtle.done()

1q.

import turtle

turtle.title("Yasir - 31")

turtle.speed(1)

turtle.begin\_fill()

turtle.color("Blue")

for \_ in range(4):

turtle.forward(100)

turtle.right(90)

turtle.end\_fill()

turtle.done()

2b.

pen.goto(100, 50)

pen.write("Rectangle", align="center", font=("Arial", 12, "normal"))

pen.penup()

pen.goto(-200, -150)

pen.pendown()

pen.setheading (45)

pen.circle (100, 90)

pen.circle (50, 90) #Half circle Other half

pen.penup()

pen.goto(-200, -200)

pen.write("Ellipse", align="center", font=("Arial", 12, "normal"))

pen.penup()

pen.goto (50, -150)

pen.pendown()

pen.setheading (0)

pen.circle (100, 180) #Half ellipse (semi-circle)

pen.penup()

pen.goto(100, -200)

pen.write("Hair Ellipse", align="center", font=("Arial", 12, "normal"))

pen.hideturtle()

turtle.done()

import turtle

turtle.title("Yasir - 31")

screen =turtle.Screen()

screen.setup(width=600, height=600)

pen=turtle.Turtle()

pen.speed (3)

pen.penup()

pen.goto (0,300)

pen.pendown ()

pen.setheading (270)

pen.forward (600)

pen.penup()

pen.goto(-300, 0 )

pen.pendown ()

pen.setheading (0)

pen. forward (600)

pen.penup()

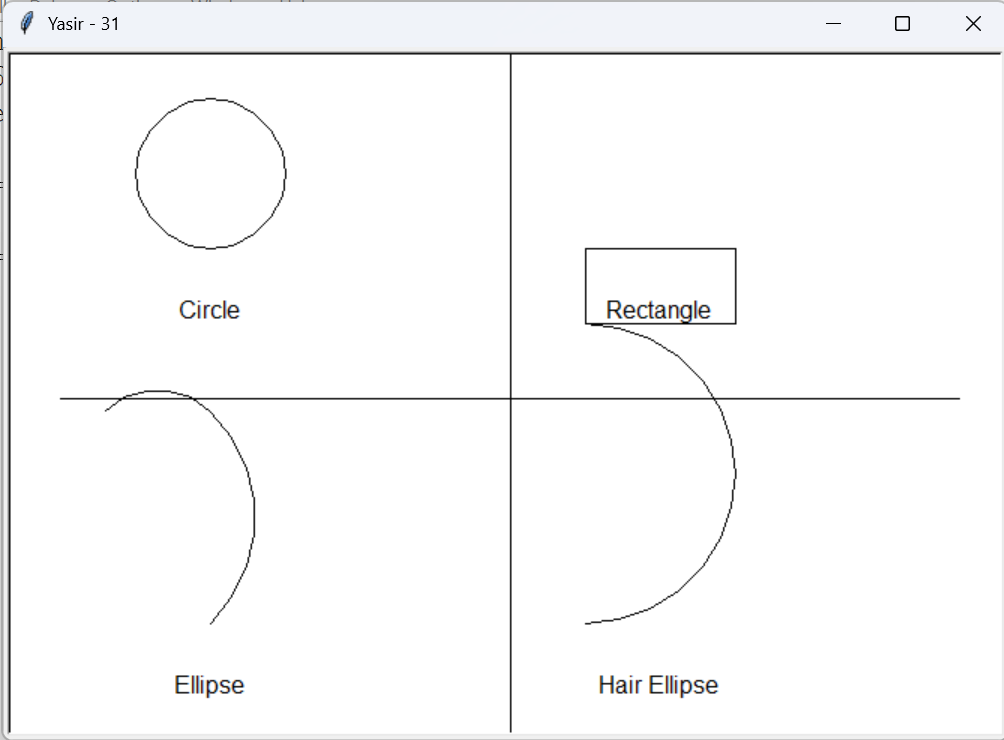
pen.goto(-200, 100)

pen.pendown()

pen.circle (50)

pen.penup()

pen.goto(-200, 50)

pen.write("Circle", align="center", font=("Arial", 12, "normal"))

pen.penup()

pen.goto(50, 100)

pen.pendown()

for \_ in range(2):

pen.forward (100)

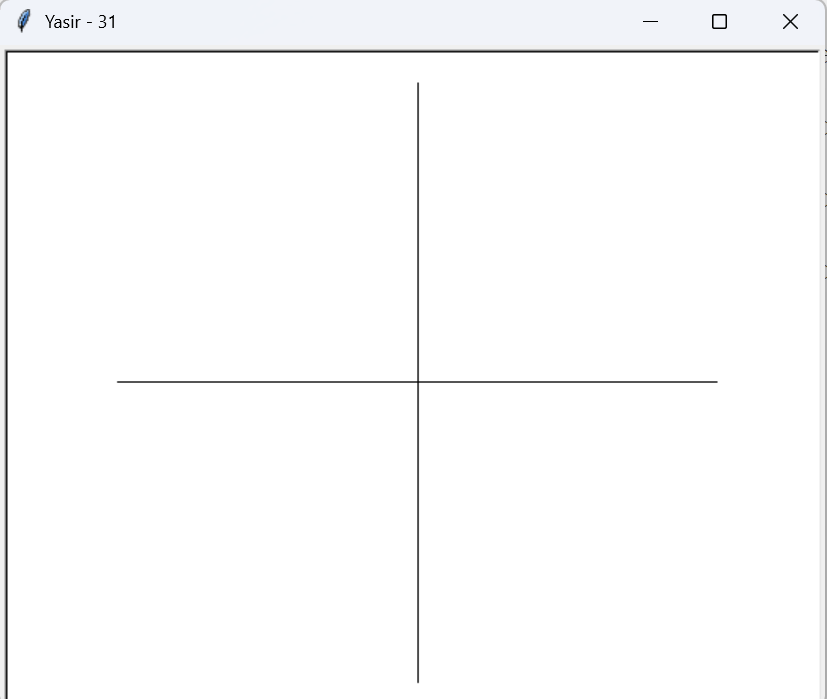
pen.right (90)

pen.forward (50)

pen.right (90)

pen.penup()

1b.

import turtle

turtle.title("Faizan-06")

axis = turtle. Turtle()

axis.speed(0)

axis.hideturtle ()

axis.penup()

axis.goto(-200, 0)

axis.pendown ()

axis.goto(200, 0)

axis.penup()

axis.goto(0, -200)

axis.pendown ()

axis.goto(0, 200)

turtle.done ()

2B.

Practical 2b:

#include <graphics.h>

#include <conio.h>

void main()

{

int gd = DETECT, gm;

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

setbkcolor(LIGHTBLUE);

cleardevice();

setcolor(WHITE);

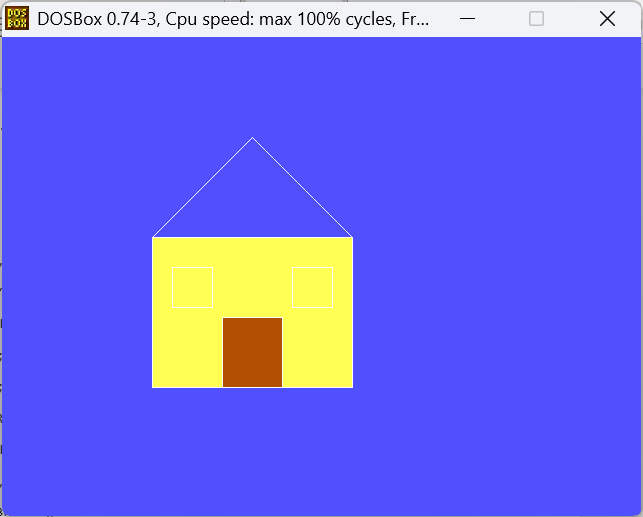
rectangle(150, 200, 350, 350);

setfillstyle(SOLID\_FILL, YELLOW);

floodfill(151, 201, WHITE);

line(150, 200, 250, 100); // Left side of the roof

line(250, 100, 350, 200); // Right side of the roof

 setfillstyle(SOLID\_FILL, RED);

floodfill(200, 150, WHITE);

rectangle(220, 280, 280, 350);

setfillstyle(SOLID\_FILL, BROWN);

floodfill(221, 281, WHITE);

rectangle(170, 230, 210, 270);

rectangle(290, 230, 330, 270);

getch();

closegraph();

}

3A.

#include<graphics.h>

#include<conio.h>

void main()

{

int gd=DETECT,gm;

int x,y,radius=80;

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

x=getmaxx()/2;

y=getmaxy()/2;

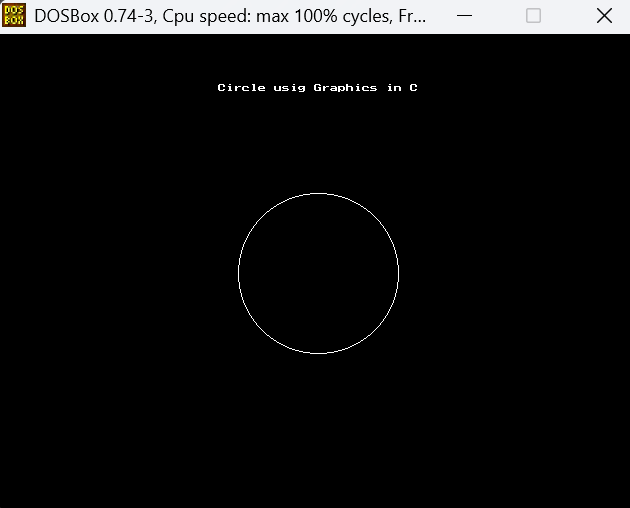
outtextxy(x-100,50,"Circle Using Graphics in C");

circle(x,y,radius);

getch();

closegraph();

}



3B.

#include<graphics.h>

#include<conio.h>

void main()

{

int gd=DETECT,gm;

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

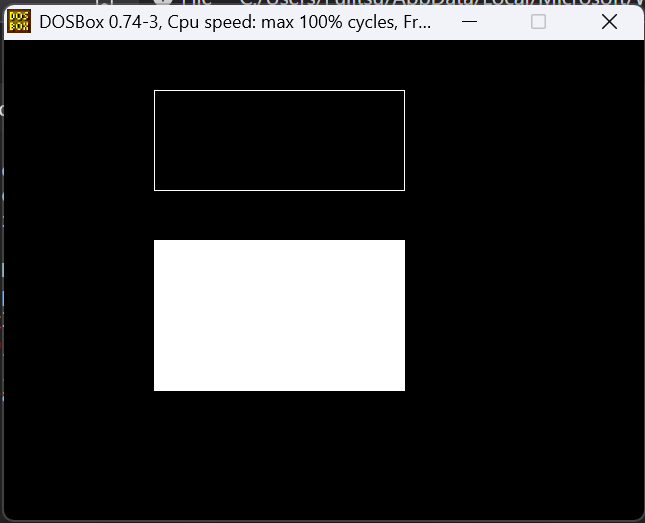
rectangle(150,50,400,150);

bar(150,200,400,350);

getch();

closegraph();

}



3C.

#include<graphics.h>

#include<conio.h>

#include<stdio.h>

#include<math.h>

void main(){

int rc,rb,xc,yc,i;

float x,y;

int gd=DETECT,gm;

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

printf("enter the radius of the outer circle\n");

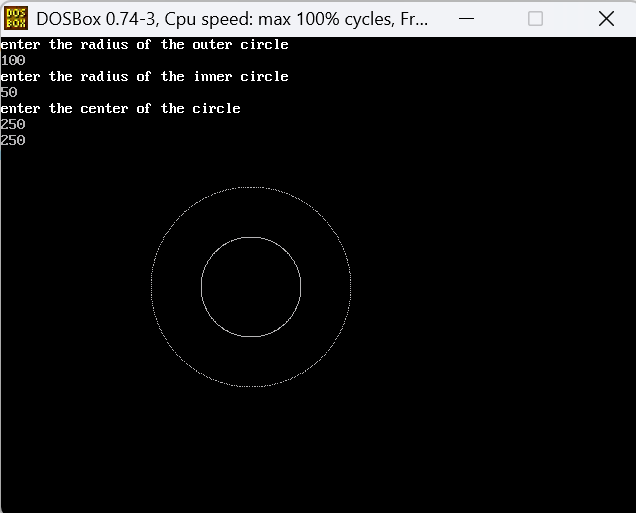
scanf("%d",&rc);

printf("enter the radius of the inner circle\n");

scanf("%d",&rb);

printf("enter the center of the circle\n");

scanf("%d",&xc);

scanf("%d",&yc);

for(i=1;i<=360;i++){

x=xc+(rb\*(cos (i)));

y=yc+(rb\*(sin (i)));

putpixel(x,y,7);

}

for(i=1;i<=360;i++)

{

x=xc+(rc\*(cos(i)));

y=yc+(rc\*(sin(i)));

putpixel(x,y,7);

}

getch();

closegraph();

}

3D.

#include<graphics.h>

#include<conio.h>

void main()

{

int gd=DETECT,gm;

int x,y;

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

x=getmaxx()/2;

y=getmaxy()/2;

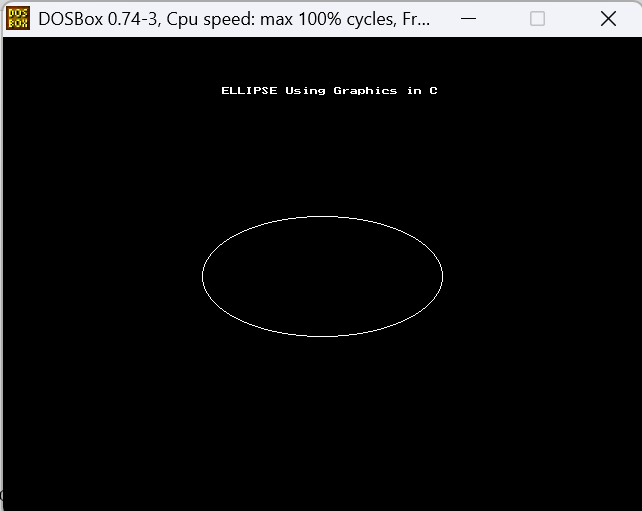
outtextxy(x-100,50,"ELLIPSE Using Graphics in C");

ellipse(x,y,0,360,120,60);

getch();

closegraph();

}



3E.

#include<graphics.h>

#include<stdio.h>

#include<conio.h>

void main()

{

int gd=DETECT,gm;

int x1=200,y1=200;

int x2=300,y2=300;

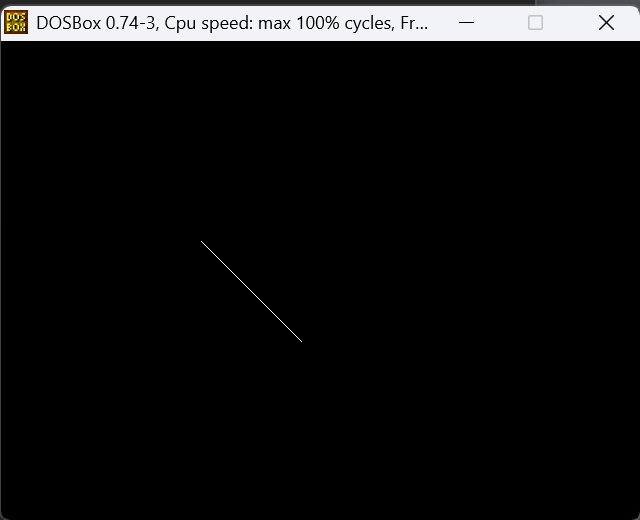
initgraph(&gd,&gm,"C:\\TURBOC3\\BGI");

line(x1,y1,x2,y2);

getch();

closegraph();

}



4A.

#include<graphics.h>

#include<stdio.h>

#include<conio.h>

#include<math.h>

#include<dos.h>

void main(){

float x,y,x1,y1,x2,y2,dx,dy,step;

int i,gd=DETECT,gm;

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

printf("enter the value of x1 and y1");

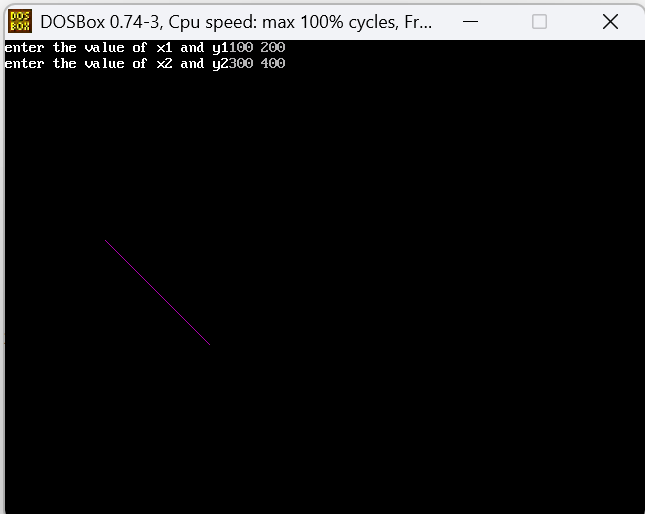
scanf("%f%f",&x1,&y1);

printf("enter the value of x2 and y2");

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

dx=abs(x2-x1); dy=abs(y2-y1);

if(dx>=dy)

step=dx;

else

step=dy;

dx=dx/step; dy=dy/step;

x=x1; y=y1; i=1;

while(i<=step){

putpixel(x,y,5);

x=x+dx; y=y+dy; i=i+1;

delay(100);

}

closegraph();

getch();

}

4B.

#include<stdio.h>

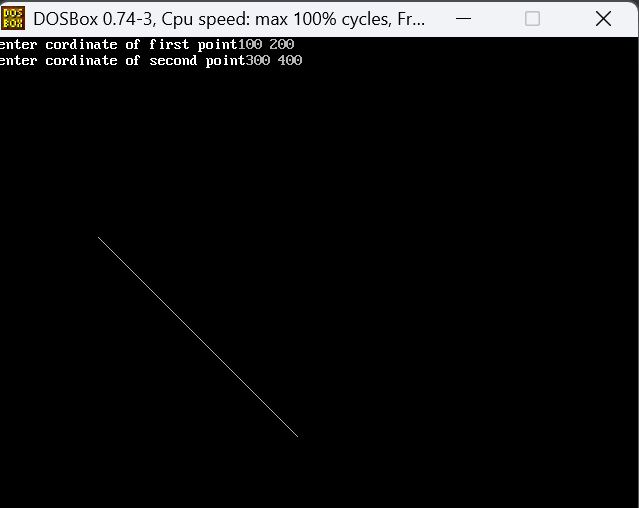
#include<conio.h>

#include<graphics.h>

void drawline(int x0,int y0,int x1,int y1)

{

int dx,dy,p,x,y;

dx=x1-x0; dy=y1-y0; x=x0; y=y0; p=2\*dy-dx;

while(x<x1){

if(p>=0){

putpixel(x,y,7);

y=y+1; p=p+2\*dy-2\*dx;

}

else{

putpixel(x,y,7);

p=p+2\*dy;

}

x=x+1;

}

}

void main(){

int gd=DETECT,gm,error,x0,y0,x1,y1;

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

printf("enter cordinate of first point");

scanf("%d%d",&x0,&y0);

printf("enter cordinate of second point");

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

drawline(x0,y0,x1,y1);

getch();

closegraph();

}

5A.

if(err<=0){

y+=1; err+=2\*y+1;

}

if(err>=0){

x-=1;

err-=2\*x+1;

}

}

}

void main(){

int gd=DETECT,gm,error,x,y,r;

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

printf("ENTER THE RADIUS OF CIRCLE:");

scanf("%d",&r);

printf("enter coordinate x and y:");

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

drawcircle(x,y,r);

getch();

closegraph();

}

#include<stdio.h>

#include<conio.h>

#include<graphics.h>

#include<dos.h>

void drawcircle(int x0,int y0,int radius)

{

int x=radius;

int y=0;

int err=0;

while(x>=y)

{

putpixel(x0+x,y0+y,7);

putpixel(x0+y,y0+x,7);

putpixel(x0-y,y0+x,7);

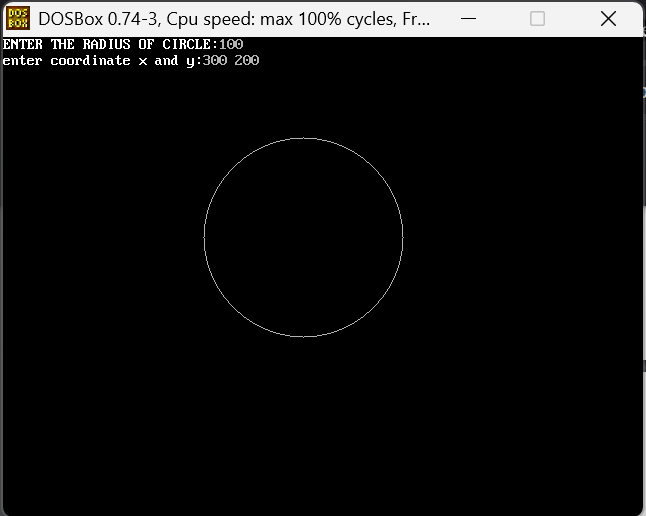
putpixel(x0-x,y0+y,7);

putpixel(x0-x,y0-y,7);

putpixel(x0-y,y0-x,7);

putpixel(x0+y,y0-x,7);

putpixel(x0+x,y0-y,7);



6A.

#include <graphics.h>

#include <stdio.h>

#include <stdlib.h>

void main()

{

int graphdriver = DETECT, graphmode;

int x1, y1, x2, y2;

int tx, ty;

int x3, y3, x4, y4;

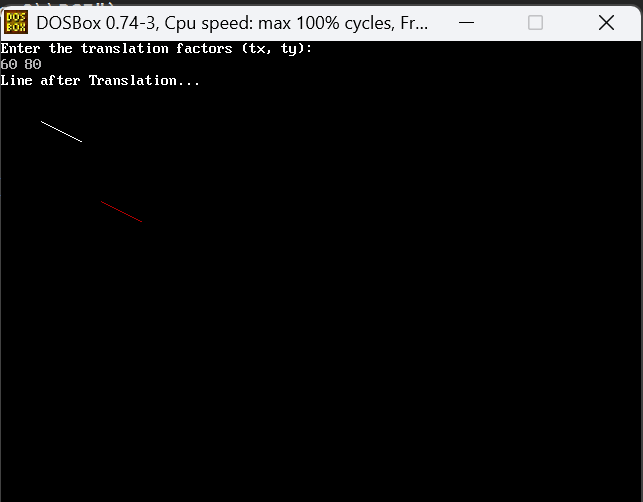
printf("Enter the coordinates of the line (x1, y1, x2, y2): \n");

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

initgraph(&graphdriver, &graphmode, "C:\\Turboc3\\BGI");

line(x1, y1, x2, y2);

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

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

x3 = x1 + tx;

y3 = y1 + ty;

x4 = x2 + tx;

y4 = y2 + ty;

printf("Line after Translation...\n");

setcolor(RED);

line(x3, y3, x4, y4);

getch();

closegraph();

}

6B.

Practical 6b:

Code:

#include <graphics.h>

#include <stdio.h>

#include <stdlib.h>

void main()

{

int graphdriver = DETECT, graphmode;

int x1, y1, x2, y2;

float sx, sy;

int x3, y3, x4, y4;

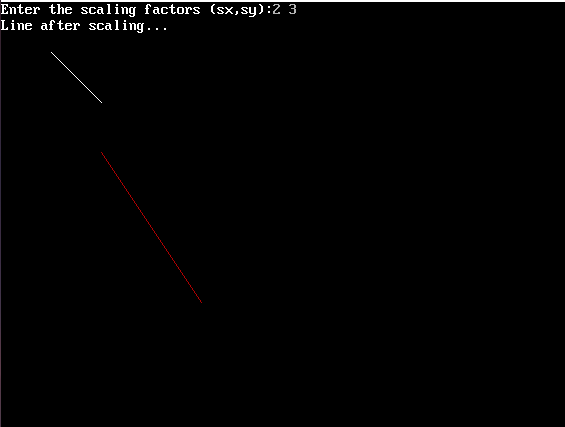
printf("Enter the coordinates of the line (x1, y1, x2, y2): ");

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

initgraph(&graphdriver, &graphmode, "C:\\TURBOC3\\BGI");

line(x1, y1, x2, y2);

printf("Enter the scaling factors (sx, sy): ");

 scanf("%f%f", &sx, &sy);

x3 = x1 \* sx;

y3 = y1 \* sy;

x4 = x2 \* sx;

y4 = y2 \* sy;

printf("Line after scaling...\n");

setcolor(RED);

line(x3, y3, x4, y4);

getch();

closegraph();

}

7A.

#include <stdlib.h>

#include <stdio.h>

#include <conio.h>

void main(){

int graphdriver = DETECT, graphmode;

int x1, y1, x2, y2, x3, y3;

int xn1, yn1, xn2, yn2, xn3, yn3;

int choice;

int midX, midY;

printf("Enter the coordinates of the triangle (x1, y1, x2, y2, x3, y3): ");

scanf("%d%d%d%d%d%d", &x1, &y1, &x2, &y2, &x3, &y3);

initgraph(&graphdriver, &graphmode, "C:\\TURBOC3\\BGI");

midX = getmaxx() / 2;

midY = getmaxy() / 2;

line(x1, y1, x2, y2);

line(x2, y2, x3, y3);

line(x3, y3, x1, y1);

printf("\nChoose the type of reflection:\n");

printf("1. Reflection over X-axis\n");

printf("2. Reflection over Y-axis\n");

printf("3. Reflection over Origin\n");

printf("Enter your choice: ");

scanf("%d", &choice);

switch (choice)

{case 1:

xn1 = x1;

yn1 = 2 \* midY - y1;

xn2 = x2;

yn2 = 2 \* midY - y2;

xn3 = x3;

yn3 = 2 \* midY - y3;

break;

case 2:

xn1 = 2 \* midX - x1;

yn1 = y1;

xn2 = 2 \* midX - x2;

yn2 = y2;

xn3 = 2 \* midX - x3;

yn3 = y3;

break;

case 3:

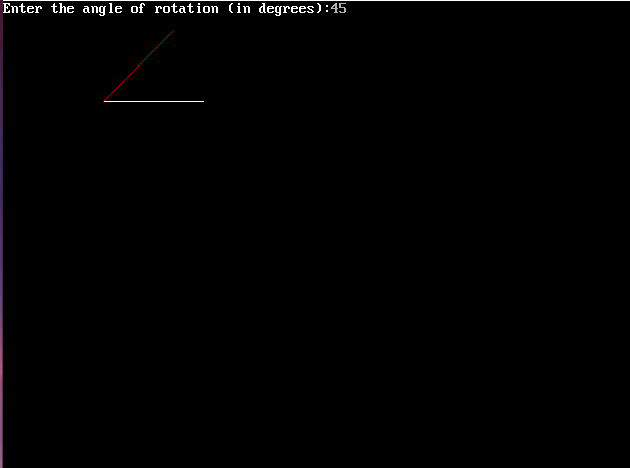
xn1 = 2 \* midX - x1;

yn1 = 2 \* midY - y1;

xn2 = 2 \* midX - x2;

yn2 = 2 \* midY - y2;

xn3 = 2 \* midX - x3;

 yn3 = 2 \* midY - y3;

break;

default:

printf("Invalid choice\n");

closegraph();

exit(0);

}

setcolor(RED);

line(xn1, yn1, xn2, yn2);

line(xn2, yn2, xn3, yn3);

line(xn3, yn3, xn1, yn1);

getch();

closegraph();

}

7B.

#include <graphics.h>

#include <stdlib.h>

#include <stdio.h>

#include <conio.h>

void main(){

int graphdriver = DETECT, graphmode;

int x1, y1, x2, y2, x3, y3;

int xn1, yn1, xn2, yn2, xn3, yn3;

int choice;

int midX, midY;

printf("Enter the coordinates of the triangle (x1, y1, x2, y2, x3, y3): ");

scanf("%d%d%d%d%d%d", &x1, &y1, &x2, &y2, &x3, &y3);

initgraph(&graphdriver, &graphmode, "C:\\TURBOC3\\BGI");

midX = getmaxx() / 2;

midY = getmaxy() / 2;

line(x1, y1, x2, y2);

line(x2, y2, x3, y3);

line(x3, y3, x1, y1);

printf("\nChoose the type of reflection:\n");

printf("1. Reflection over X-axis\n");

printf("2. Reflection over Y-axis\n");

printf("3. Reflection over Origin\n");

printf("Enter your choice: ");

scanf("%d", &choice);

switch (choice)

{case 1:

xn1 = x1;

yn1 = 2 \* midY - y1;

xn2 = x2;

yn2 = 2 \* midY - y2;

xn3 = x3;

yn3 = 2 \* midY - y3;

break;

case 2:

xn1 = 2 \* midX - x1;

yn1 = y1;

xn2 = 2 \* midX - x2;

yn2 = y2;

xn3 = 2 \* midX - x3;

yn3 = y3;

break;

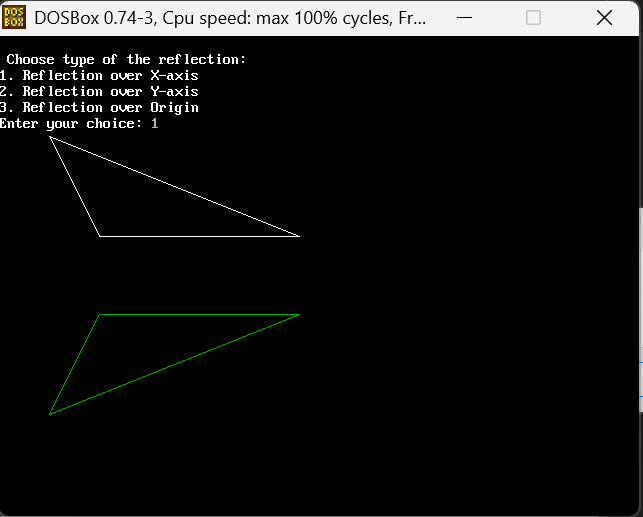
case 3:

xn1 = 2 \* midX - x1;

yn1 = 2 \* midY - y1;

xn2 = 2 \* midX - x2;

yn2 = 2 \* midY - y2;

 xn3 = 2 \* midX - x3;

yn3 = 2 \* midY - y3;

break;

default:

printf("Invalid choice\n");

closegraph();

exit(0);

}

setcolor(RED);

line(xn1, yn1, xn2, yn2);

line(xn2, yn2, xn3, yn3);

line(xn3, yn3, xn1, yn1);

getch();

closegraph();

}

7C.

#include <graphics.h>

#include <stdlib.h>

#include <stdio.h>

#include <conio.h>

void main(){

int graphdriver = DETECT, graphmode;

int x1, y1, x2, y2, x3, y3;

int xn1, yn1, xn2, yn2, xn3, yn3;

int choice;

int midX, midY;

printf("Enter the coordinates of the triangle (x1, y1, x2, y2, x3, y3): ");

scanf("%d%d%d%d%d%d", &x1, &y1, &x2, &y2, &x3, &y3);

initgraph(&graphdriver, &graphmode, "C:\\TURBOC3\\BGI");

midX = getmaxx() / 2;

midY = getmaxy() / 2;

line(x1, y1, x2, y2);

line(x2, y2, x3, y3);

line(x3, y3, x1, y1);

printf("\nChoose the type of reflection:\n");

printf("1. Reflection over X-axis\n");

printf("2. Reflection over Y-axis\n");

printf("3. Reflection over Origin\n");

printf("Enter your choice: ");

scanf("%d", &choice);

switch (choice)

{case 1:

xn1 = x1;

yn1 = 2 \* midY - y1;

xn2 = x2;

yn2 = 2 \* midY - y2;

xn3 = x3;

yn3 = 2 \* midY - y3;

break;

case 2:

xn1 = 2 \* midX - x1;

yn1 = y1;

xn2 = 2 \* midX - x2;

yn2 = y2;

xn3 = 2 \* midX - x3;

yn3 = y3;

break;

case 3:

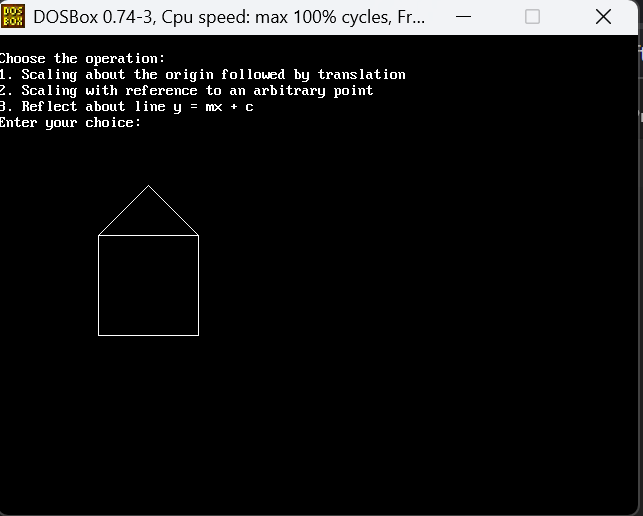
xn1 = 2 \* midX - x1;

yn1 = 2 \* midY - y1;

xn2 = 2 \* midX - x2;

yn2 = 2 \* midY - y2;

xn3 = 2 \* midX - x3;

 yn3 = 2 \* midY - y3;

break;

default:

printf("Invalid choice\n");

closegraph();

exit(0);

}

setcolor(RED);

line(xn1, yn1, xn2, yn2);

line(xn2, yn2, xn3, yn3);

line(xn3, yn3, xn1, yn1);

getch();

closegraph();

}

8A.

#include <graphics.h>

#include <conio.h>

#include <stdio.h>

#define TOP 8

#define BOTTOM 4

#define RIGHT 2

#define LEFT 1

int xmin, ymin, xmax, ymax;

int computeCode(int x, int y) {

int code = 0;

if (y > ymax) code |= TOP;

if (y < ymin) code |= BOTTOM;

if (x > xmax) code |= RIGHT;

if (x < xmin) code |= LEFT;

return code;

}

void cohenSutherlandClip(int x0, int y0, int x1, int y1) {

int code0 = computeCode(x0, y0);

int code1 = computeCode(x1, y1);

int codeOut;

int accept = 0;

float x, y;

while (1) {

if ((code0 == 0) && (code1 == 0)) { // Both endpoints inside

accept = 1;

break;

} else if (code0 & code1) { // Both endpoints share an outside region (completely

outside)

break;

} else {

codeOut = code0 ? code0 : code1;

if (codeOut & TOP) {

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

y = ymax;

} else if (codeOut & BOTTOM) {

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

y = ymin;

} else if (codeOut & RIGHT) {

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

x = xmax;

} else if (codeOut & LEFT) {

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

x = xmin;

}

if (codeOut == code0) {

x0 = (int)x;

y0 = (int)y;

code0 = computeCode(x0, y0);

} else {

x1 = (int)x;

y1 = (int)y;

code1 = computeCode(x1, y1);

}

}

}

if (accept) {

setcolor(GREEN);

line(x0, y0, x1, y1);

}

}

void main() {

int gd = DETECT, gm;

int x0, y0, x1, y1;

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

printf("Enter clipping window (xmin ymin xmax ymax): ");

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

printf("Enter line coordinates (x0 y0 x1 y1): ");

scanf("%d %d %d %d", &x0, &y0, &x1, &y1);

// Draw clipping window

setcolor(WHITE);

rectangle(xmin, ymin, xmax, ymax);

setcolor(RED);

line(x0, y0, x1, y1);

getch();

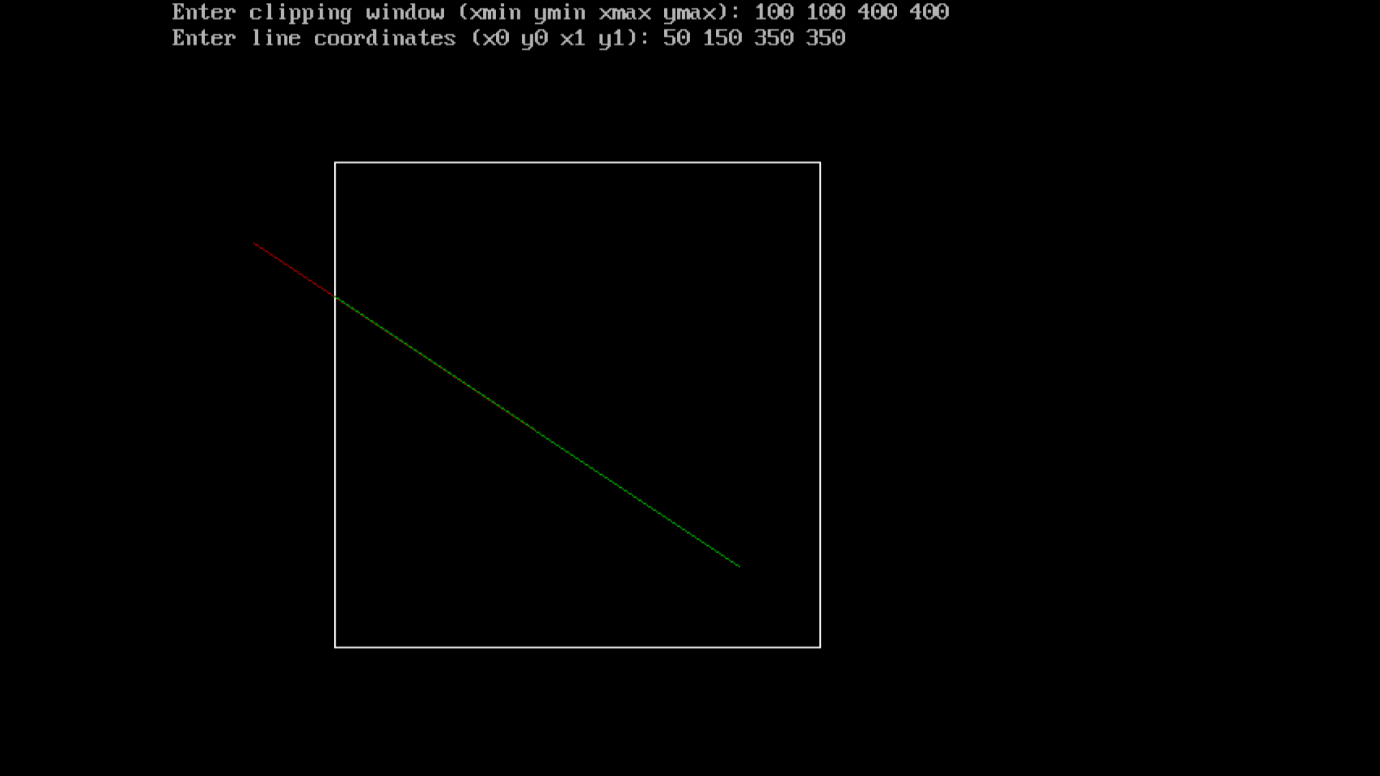
// Perform clipping and draw the result

cohenSutherlandClip(x0, y0, x1, y1);

getch();

closegraph();

}



9A.

#include <graphics.h>

#include <stdio.h>

#include <conio.h>

#include <dos.h>

void customFloodFill(int x, int y, int oldcolor, int newcolor) {

if (getpixel(x, y) == oldcolor) {

delay(20);

putpixel(x, y, newcolor);

customFloodFill(x + 1, y, oldcolor, newcolor);

customFloodFill(x - 1, y, oldcolor, newcolor);

customFloodFill(x, y + 1, oldcolor, newcolor);

customFloodFill(x, y - 1, oldcolor, newcolor);

}

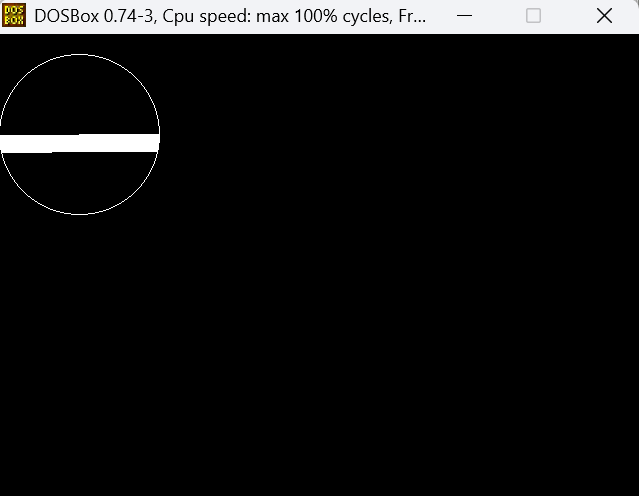
}

void main() {

int gd = DETECT, gm;

int x, y, radius;

printf("Enter x and y position for circle: ");

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

printf("Enter radius of circle: ");

scanf("%d", &radius);

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

circle(x, y, radius);

customFloodFill(x, y, BLACK, WHITE);

getch();

closegraph();

}

9B.

#include <graphics.h>

#include <stdio.h>

#include <conio.h>

#include <dos.h>

void boundaryFill(int x, int y, int fillColor, int boundaryColor) {

if (getpixel(x, y) != boundaryColor && getpixel(x, y) != fillColor) {

delay(20);

putpixel(x, y, fillColor);

boundaryFill(x + 1, y, fillColor, boundaryColor);

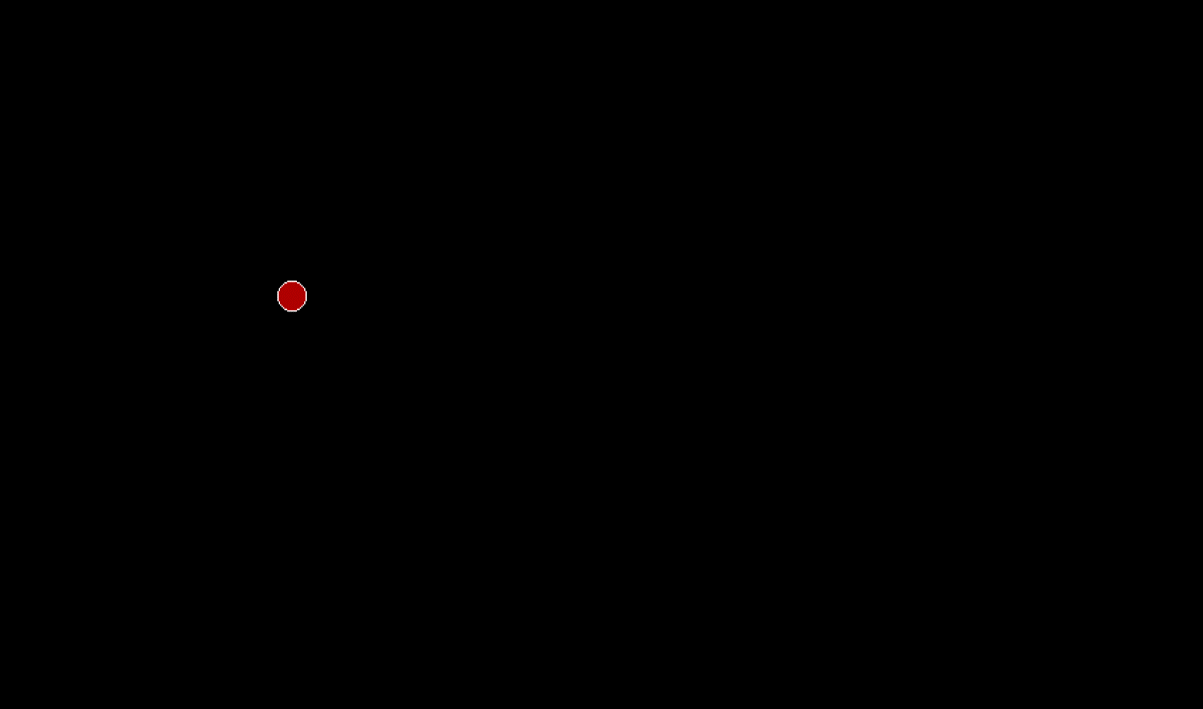
boundaryFill(x - 1, y, fillColor, boundaryColor);

boundaryFill(x, y + 1, fillColor, boundaryColor);

boundaryFill(x, y - 1, fillColor, boundaryColor);

}

}

void main() {

int gd = DETECT, gm;

int x, y, radius;

printf("Enter x and y position for circle: ");

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

printf("Enter radius of circle: ");

scanf("%d", &radius);

initgraph(&gd, &gm, "C:\\Turb

circle(x, y, radius);

boundaryFill(x, y, RED, WHITE);

getch();

closegraph();

}

10A.

#include<graphics.h>

#include<conio.h>

#include<stdio.h>

#include<dos.h>

void main()

{

int gd=DETECT,gm,i,maxx,maxy,key0;

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

maxx=getmaxx();

maxy=getmaxy();

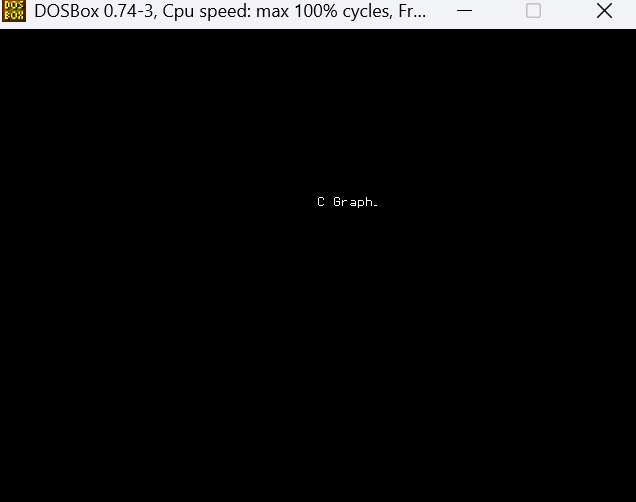
while(!kbhit())

{

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

{

cleardevice();

settextstyle(2,0,5);

outtextxy(maxx/2,i,"C Graphics");

delay(100);

}

}

getch();

}

10B.

#include<graphics.h>

#include<conio.h>

#include<stdio.h>

void main()

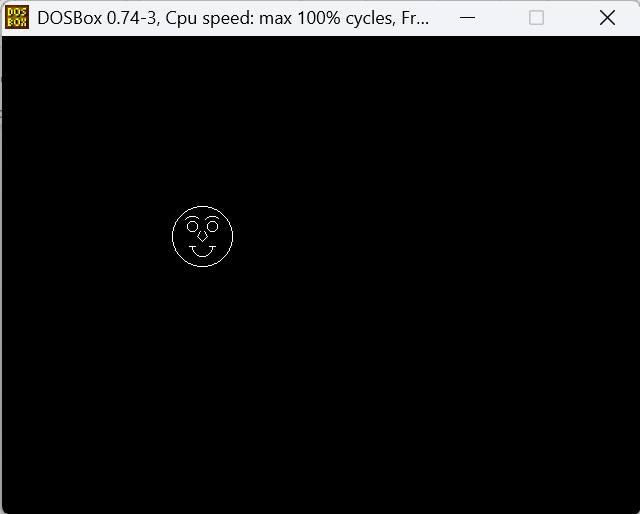
{

int gd=DETECT,gm;

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

circle(200,200,30);

circle(190,190,5);

arc(190,190,50,130,10);

circle(210,190,5);

arc(210,190,50,130,10);

arc(200,210,180,360,10);

line(187,210,193,210);

line(207,210,213,210);

line(198,195,195,200);

line(202,195,205,200);

line(195,200,200,205);

line(205,200,200,205);

getch();

closegraph();

}

10C.

#include<graphics.h>

#include <graphics.h>

#include <dos.h>

#include <conio.h>

void main() {

int i, j = 0, gd = DETECT, gm;

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

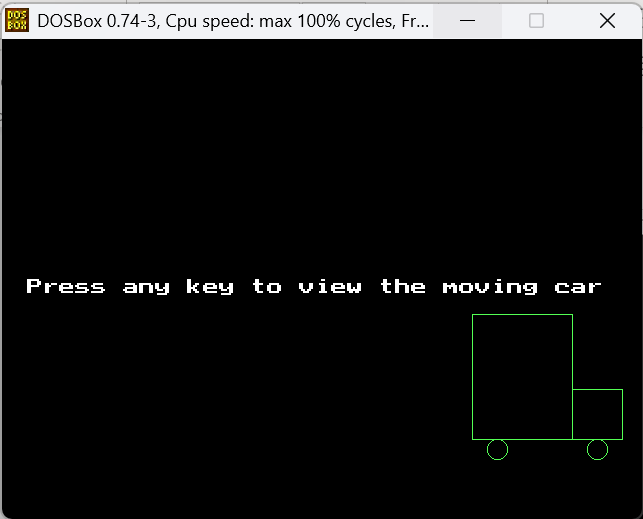
settextstyle(DEFAULT\_FONT, HORIZ\_DIR, 2);

outtextxy(25, 240, "Press any key to view the moving car");

getch();

for (i = 0; i <= 420; i += 10, j++) {

setcolor(j % 16);

 rectangle(50 + i, 275, 150 + i, 400);

rectangle(150 + i, 350, 200 + i, 400);

circle(75 + i, 410, 10);

circle(175 + i, 410, 10);

delay(100);

if (i < 420) {

setcolor(BLACK);

rectangle(50 + i, 275, 150 + i, 400);

rectangle(150 + i, 350, 200 + i, 400);

circle(75 + i, 410, 10);

circle(175 + i, 410, 10);

}

}

getch();

closegraph();

}