PROGRAM - 1

Q – Write a program for 2-D line drawing as Raster Graphics Display.

Code:

#include <iostream.h>

#include <conio.h>

#include <graphics.h>

#include <math.h>

#define ROUND(a) ((int) (a+0.5))

void lineDDA(int xa, int xb, int ya, int yb)

{ int dx=xb-xa, dy=yb-ya, steps, k;

float xinc, yinc, x=xa, y=ya;

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

steps=abs(dx);

else

steps=abs(dy);

xinc=dx/(float)steps;

yinc=dy/(float)steps;

putpixel(ROUND(x),ROUND(y),RED);

for(k=0; k<steps; k++)

{ x+=xinc;

y+=yinc;

putpixel(ROUND(x),ROUND(y),RED);

}

}

void main()

{

int gd=DETECT,gm,x,y;

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

setbkcolor(WHITE);

lineDDA(50,400,50,200);

getch();

closegraph();

}

OUTPUT:



PROGRAM - 3

Q – Write a program to draw an ellipse using Midpoint algorithm.

Code:

#include<conio.h>

#include<iostream.h>

#include<graphics.h>

#define ROUND(a) ((int) a+0.5)

void ellipseMidpoint(int xcenter, int ycenter, int rx, int ry)

{

int rx2 = rx\*rx;

int ry2 = ry\*ry;

int tworx2 = 2\*rx2;

int twory2 = 2\*ry2;

int p;

int x=0;

int y=ry;

int px=0;

int py = tworx2 \* y;

void ellipsePlotpoints(int, int, int, int);

ellipsePlotpoints(xcenter, ycenter, x, y);

p=ROUND(ry2-(rx2\*ry)+(0.25\*rx2));

while(px<py){

x++;

px += twory2;

if(p<0)

p+=ry2+px;

else {

y--;

py -=tworx2;

p +=ry2 +px-py;

}

ellipsePlotpoints(xcenter, ycenter, x, y);

}

p=ROUND(ry2\*(x+0.5)\*(x+0.5) + rx2\*(y-1)\*(y-1)- rx2\*ry2);

while(y>0){

y--;

py -= tworx2;

if(p>0)

p+=rx2-py;

else{

x++;

px+=twory2;

p+=rx2-py+px;

}

ellipsePlotpoints(xcenter, ycenter, x, y);

}

}

void ellipsePlotpoints(int xcenter, int ycenter, int x, int y)

{

putpixel(xcenter+x, ycenter+y, RED);

putpixel(xcenter-x, ycenter+y, RED);

putpixel(xcenter+x, ycenter-y, RED);

putpixel(xcenter-x, ycenter-y, RED);

}

int main()

{

int gmode, gdriver=DETECT;

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

setbkcolor(WHITE);

ellipseMidpoint(300,300,24,18);

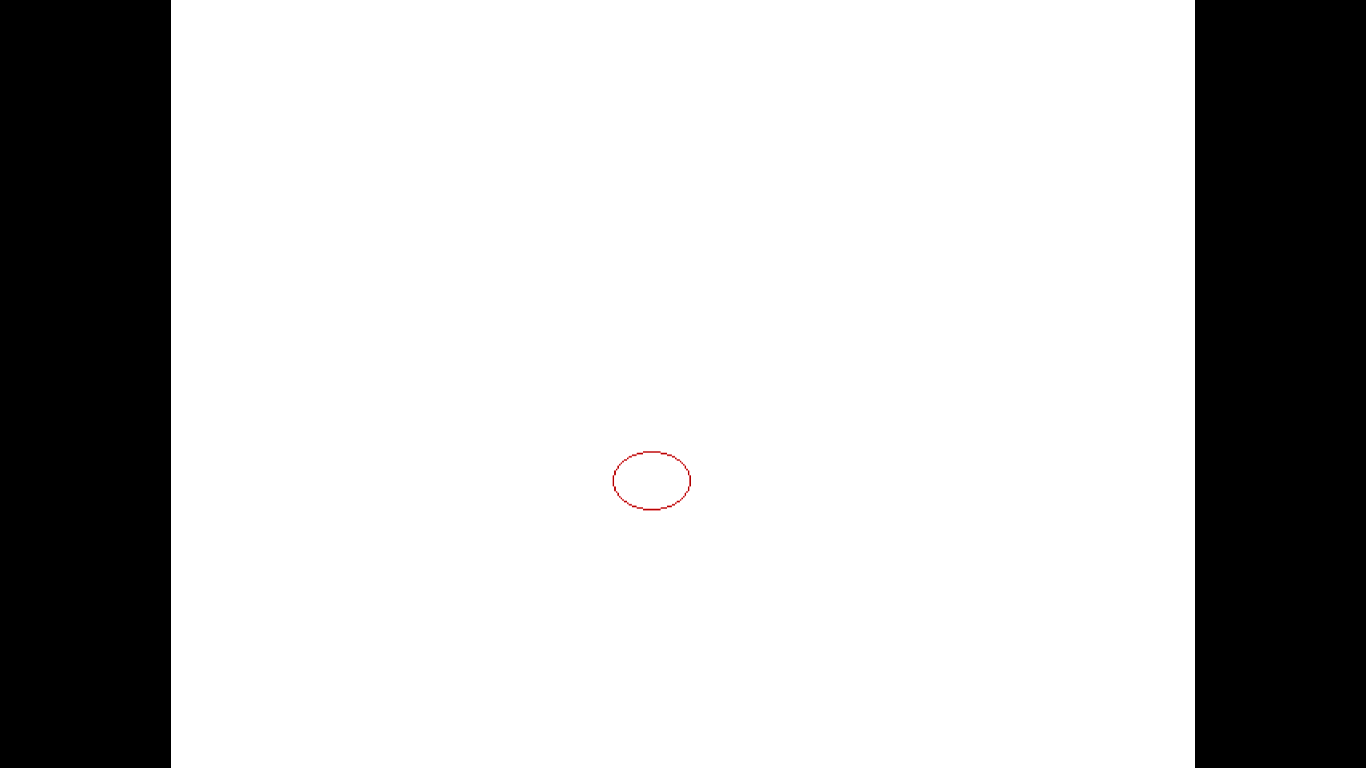
getch();

closegraph();

return 0;

}

OUTPUT:



PROGRAM – 2

Q - Write a program to draw a circle using Midpoint Algorithm. Modify the same for drawing arc and sector.

Code:

#include<iostream.h>

#include<graphics.h>

#include<conio.h>

void circleMidpoint(int xCenter, int yCenter, int radius)

{

int x=0;

int y=radius;

int p=1-radius;

void circlePlotPoints(int, int, int, int);

circlePlotPoints(xCenter, yCenter, x, y);

while(x<y){

x++;

if(p<0)

p += 2\*x+1;

else{

y--;

p += 2\*(x-y)+1;

}

circlePlotPoints(xCenter, yCenter, x, y);

}

}

void circlePlotPoints(int xCenter, int yCenter, int x, int y)

{

putpixel(xCenter + x, yCenter + y, RED);

putpixel(xCenter - x, yCenter + y, RED);

putpixel(xCenter + x, yCenter - y, RED);

putpixel(xCenter - x, yCenter - y, RED);

putpixel(xCenter + y, yCenter + x, RED);

putpixel(xCenter - y, yCenter + x, RED);

putpixel(xCenter + y, yCenter - x, RED);

putpixel(xCenter - y, yCenter - x, RED);

}

int main()

{

int gdriver=DETECT, gmode;

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

circleMidpoint(300,300,100);

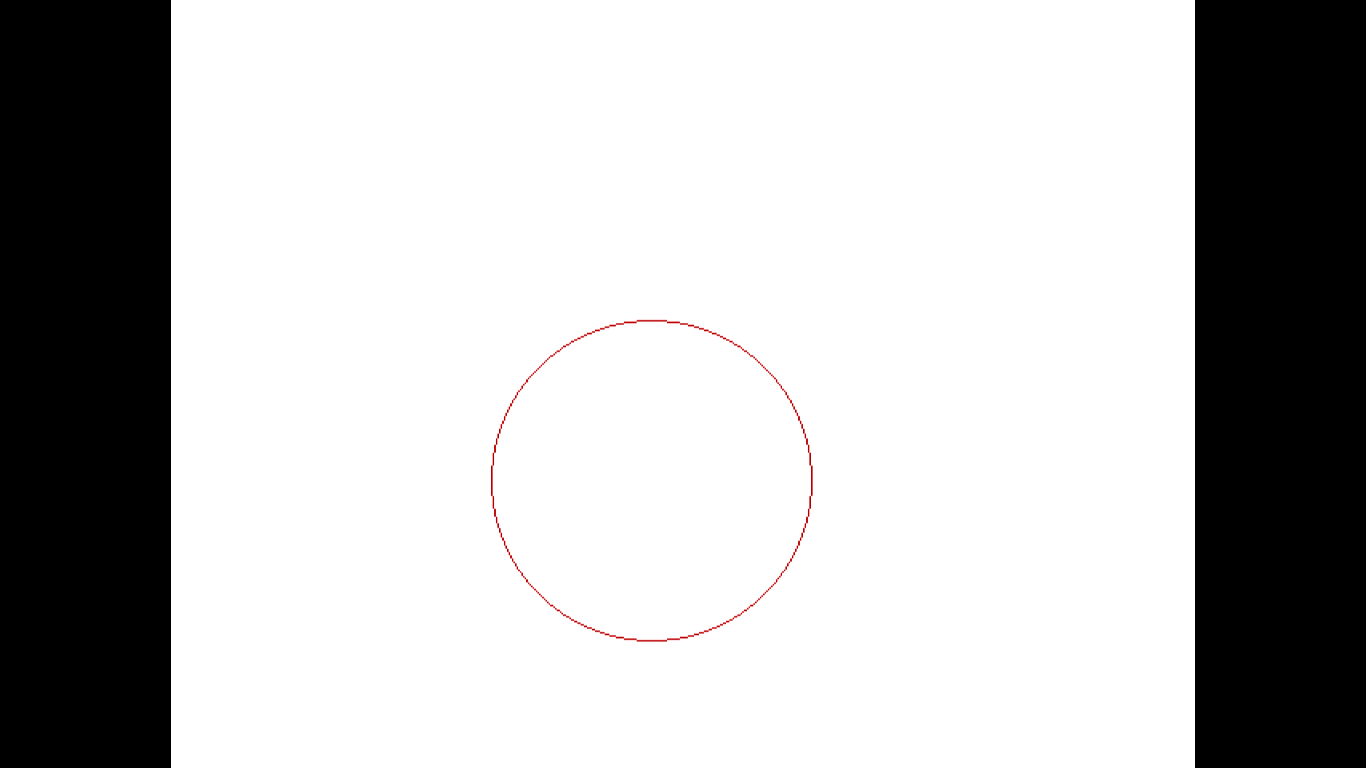
getch();

closegraph();

return 0;

}

OUTPUT:



NETAJI SUBHAS INSTITUTE OF TECHNOLOGY



COMPUTER GRAPHICS

ITC09

ADITI GUPTA

2016UIT2588

INDEX

|  |  |  |  |
| --- | --- | --- | --- |
| S.NO. | PROGRAM | DATE | SIGN |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

PROGRAM – 10

Q- Write a program for line clipping.

Code: (Cohen-Sutherland)

#include<iostream.h>

#include<graphics.h>

#include<conio.h>

#define LEFT 0x01

#define RIGHT 0x4

#define BOTTOM 0x2

#define TOP 0x8

char getcode(float x,float y,float xwmin,float ywmin,float xwmax,float ywmax)

{

unsigned char code = 0x00;

if(x<xwmin)

code=code|LEFT;

if(x>xwmax)

code=code|RIGHT;

if(y>ywmin)

code=code|BOTTOM;

if(y<ywmax)

code=code|TOP;

return code;

}

void clipLine(float x1,float y1,float x2,float y2,float xwmin,float ywmin,float xwmax,float ywmax)

{

int done = 0,accept=0;

unsigned char code1,code2;

setcolor(BLUE);

line(300,0,300,479);

setcolor(RED);

line(0,240,639,240);

setcolor(YELLOW);

rectangle(xwmin,ywmin,xwmax,ywmax);

setcolor(GREEN);

line(x1,y1,x2,y2);

getch();

setcolor(WHITE);

float m;

while(done==0)

{

code1=getcode(x1,y1,xwmin,ywmin,xwmax,ywmax);

code2=getcode(x2,y2,xwmin,ywmin,xwmax,ywmax);

/\*case 1 - accept line\*/

if(((code1&code2)==0)&&((code1|code2)==0))

{

accept = 1;

done = 1;

}

else if((code1&code2)!=0)

{

done=1;

outtextxy(10,300,"\nline rejected");

}

else

{

if((x1>xwmin&&x1<=xwmax)&&(y1>=ywmax&&y1<=ywmin))

{ float temp=x1;

x1 = x2;

x2=temp;

temp = y1;

y1=y2;

y2=temp;

char t;

t=code1;

code1=code2;

code2=t;

}

if(x1!=x2)

m = (y2-y1)/(x2-x1);

if( code1 & LEFT != 0)

{

y1+= (xwmin-x1)\*m;

x1 = xwmin;

}

else if(code1 & RIGHT)

{

y1+= (xwmax-x1)\*m;

x1 = xwmax;

}

else if(code1 & BOTTOM)

{

if(x2!=x1)

x1+= (ywmin - y1)/m;

y1 = ywmin;

}

else

{

if(x2!=x1)

x1+= (ywmax-y1)/m;

y1 = ywmax;

}

}

}

if(accept == 1)

line(x1,y1,x2,y2);

}

void main()

{

int gd=DETECT,gmode;

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

float xwmin, xwmax, ywmin, ywmax;

cout << "Enter the x limits for the clipping window : ";

cin >> xwmin >> xwmax;

cout <<"\n Enter the y limits fro the clipping window :";

cin >> ywmin >> ywmax;

cout<<"\n Enter end point 1 : ";

float x1,y1,x2,y2;

cin>>x1>>y1;

cout<<"\n Enter end point 2 : ";

cin>>x2>>y2;

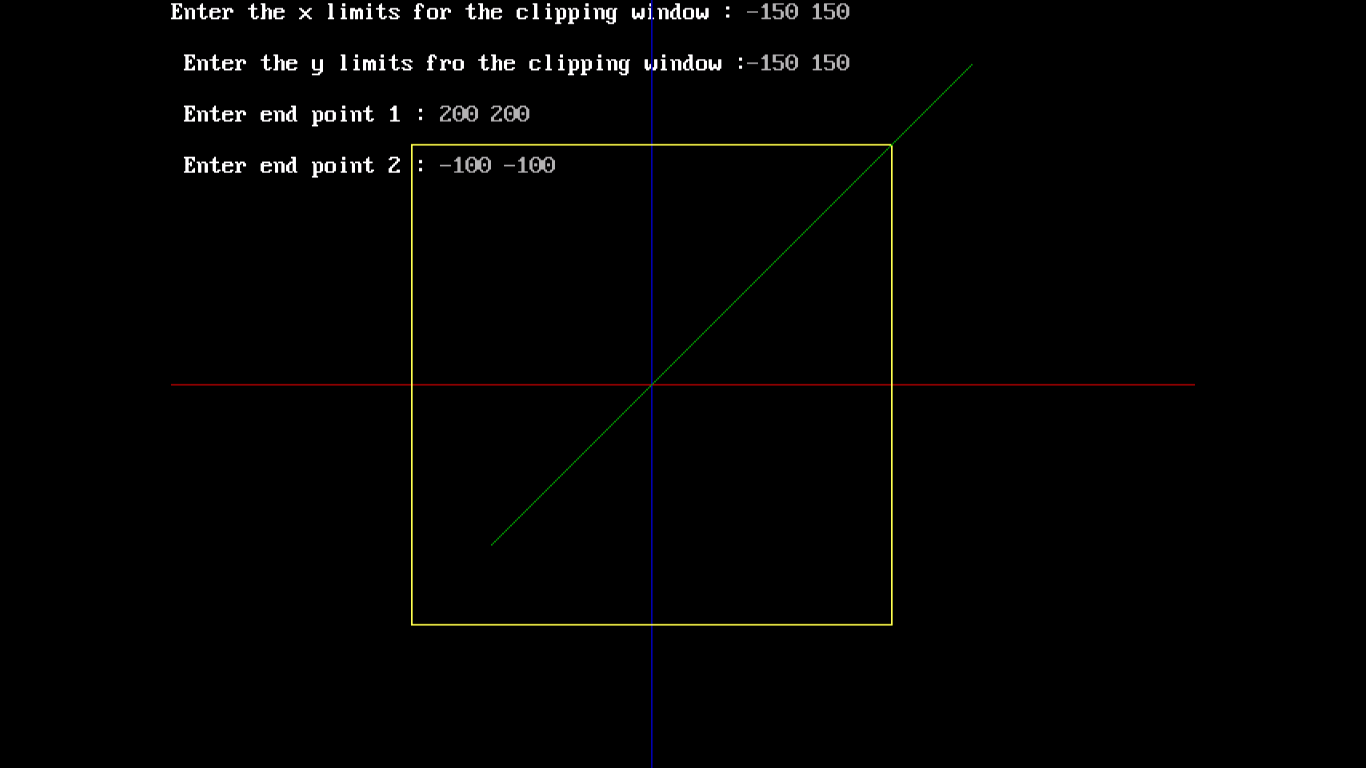
clipLine(x1+300,240-y1,x2+300,240-y2,xwmin+300,240-ywmin,xwmax+300,240-ywmax);

while(!kbhit()) {}

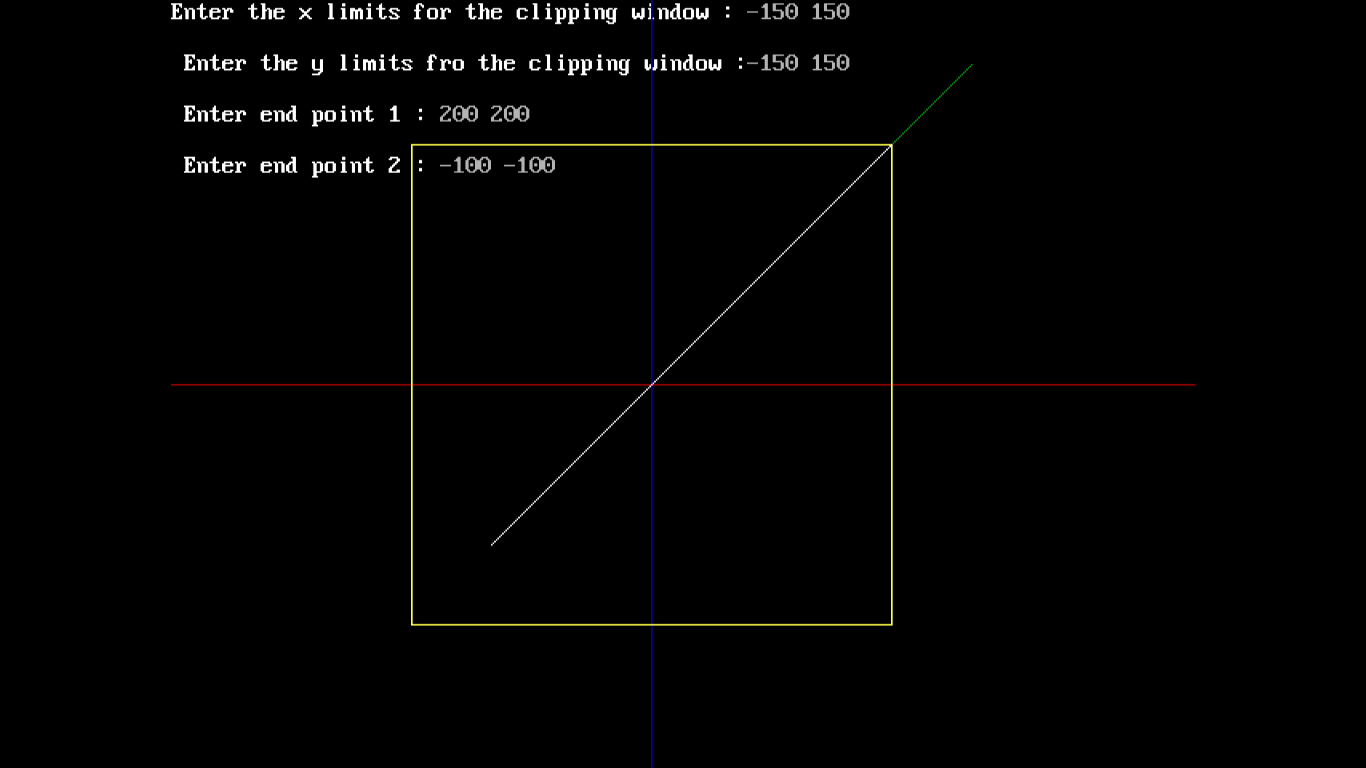
closegraph(); }

OUTPUT:

Before Clipping



After Clipping



PROGRAM – 11

Q – Write a program for polygon clipping.

CODE:

#include<iostream.h>

#include<conio.h>

#include<graphics.h>

struct point

{

int x,y;

};

int createlist(int i, point cl[], int nc, point s[], int ns)

{

point t[20];

int k=0;

if(i==0) // TOP EDGE

{

for(int j=0;j<ns;j++)

{

// o -> i

if(s[j].y<cl[0].y && s[j+1].y>cl[0].y)

{

//find point of intersection

int ax = int(((cl[0].y-s[j].y)\*(s[j+1].x-s[j].x)/(s[j+1].y-s[j].y)\*1.0)+s[j].x);

t[k].x = ax;

t[k].y = cl[0].y;

k++;

t[k] = s[j+1];

k++;

}

//i -> o

else if(s[j].y>cl[0].y && s[j+1].y<cl[0].y)

{

int ax = int(((cl[0].y-s[j].y)\*(s[j+1].x-s[j].x)/(s[j+1].y-s[j].y)\*1.0)+s[j].x);

t[k].x = ax;

t[k].y = cl[0].y;

k++;

}

//i -> i

else if(s[j].y>cl[0].y && s[j+1].y>cl[0].y)

{

t[k] = s[j+1];

k++;

}

//o -> o => do nothing

}

}

else if(i==1) // RIGHT EDGE

{

for(int j=0;j<ns;j++)

{

// o -> i

if(s[j].x>cl[1].x && s[j+1].x<cl[1].x)

{

//find point of intersection

int ay = int(((cl[1].x-s[j].x)\*(s[j+1].y-s[j].y)/(s[j+1].x-s[j].x)\*1.0)+s[j].y);

t[k].x = cl[1].x;

t[k].y = ay;

k++;

t[k] = s[j+1];

k++;

}

//i -> o

else if(s[j].x<cl[1].x && s[j+1].x>cl[1].x)

{

int ay = int(((cl[1].x-s[j].x)\*(s[j+1].y-s[j].y)/(s[j+1].x-s[j].x)\*1.0)+s[j].y);

t[k].x = cl[1].x;

t[k].y = ay;

k++;

}

//i -> i

else if(s[j].x<cl[1].x && s[j+1].x<cl[1].x)

{

t[k] = s[j+1];

k++;

}

//o -> o => do nothing

}

}

else if(i==2) // BOTTOM EDGE

{

for(int j=0;j<ns;j++)

{

// o -> i

if(s[j].y>cl[2].y && s[j+1].y<cl[2].y)

{

//find point of intersection

int ax = int(((cl[2].y-s[j].y)\*(s[j+1].x-s[j].x)/(s[j+1].y-s[j].y)\*1.0)+s[j].x);

t[k].x = ax;

t[k].y = cl[2].y;

k++;

t[k] = s[j+1];

k++;

}

//i -> o

else if(s[j].y<cl[2].y && s[j+1].y>cl[2].y)

{

int ax = int(((cl[2].y-s[j].y)\*(s[j+1].x-s[j].x)/(s[j+1].y-s[j].y)\*1.0)+s[j].x);

t[k].x = ax;

t[k].y = cl[2].y;

k++;

}

//i -> i

else if(s[j].y<cl[2].y && s[j+1].y<cl[2].y)

{

t[k] = s[j+1];

k++;

}

//o -> o => do nothing

}

}

else if(i==3) // LEFT EDGE

{

for(int j=0;j<ns;j++)

{

// o -> i

if(s[j].x<cl[0].x && s[j+1].x>cl[0].x)

{

//find point of intersection

int ay = int(((cl[0].x-s[j].x)\*(s[j+1].y-s[j].y)/(s[j+1].x-s[j].x)\*1.0)+s[j].y);

t[k].x = cl[0].x;

t[k].y = ay;

k++;

t[k] = s[j+1];

k++;

}

//i -> o

else if(s[j].x>cl[0].x && s[j+1].x<cl[0].x)

{

int ay = int(((cl[0].x-s[j].x)\*(s[j+1].y-s[j].y)/(s[j+1].x-s[j].x)\*1.0)+s[j].y);

t[k].x = cl[0].x;

t[k].y = ay;

k++;

}

//i -> i

else if(s[j].x>cl[0].x && s[j+1].x>cl[0].x)

{

t[k] = s[j+1];

k++;

}

//o -> o => do nothing

}

}

t[k]=t[0];

for(int l=0;l<=k;l++)

{

s[l]=t[l];

}

return k;

}

/\*int is\_out(point p, point cl[], int nc)

{

// 1. I have taken clockwise orientation positive

// 2. So any point is inside the polygon if it is to the left of every edge

// 3. Otherwise it is outside.

// 4. Let edge be p1p2(p1 & p2 in clockwise order). Let point be P.

// therefore, if slope(p1p2) > slope(p1P), for every edge,

// then the point is inside the polygon

int in = 0;

int out = 0;

int i=0;

while (i<nc && out==0)

{

dec = ((c[i+1].y - c[i].y) \* (p.x - c[i].x)) - ((p.y - c[i].y) \* (c[i+1].x - c[i].x));

if(dec < 0)

out =1;

i++;

}

return out;

if((p.x >= cl[0].x)&&(p.x <= cl[1].x)&&(p.y >= cl[0].y)&&(p.y <= cl[2].y))

return 0;

else

return 1;

} \*/

void suthodg( point cl[], int nc, point s[], int ns)

{

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

{

ns = createlist(i, cl, nc, s, ns);

}

setcolor(GREEN);

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

{

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

}

}

void main()

{

clrscr();

point cl[4];

point sub[10], dup\_sub[10];

int nc;

cout<<" Clipping Polygon ";

//min is top left and max is bottom right

cout<<"\n Enter the co-ordinates (x,y) ";

cout<<"\n Xwmin : ";

cin>>cl[0].x;

cout<<"\n Ywmin : ";

cin>>cl[0].y;

cout<<"\n Xwmax : ";

cin>>cl[2].x;

cout<<"\n Ywmax : ";

cin>>cl[2].y;

cl[1].x = cl[2].x;

cl[1].y = cl[0].y;

cl[3].x = cl[0].x;

cl[3].y = cl[2].y;

int ns;

cout<<" Subject Polygon ";

do

{

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

cin>>ns;

}while(ns>10);

cout<<"\n Enter the co-ordinates in clockwise order (x,y) ";

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

{

cout <<i+1<<". ";

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

dup\_sub[i] = sub[i];

}

sub[i] = sub[0];

dup\_sub[i] = sub[0];

int gdriver = DETECT, gmode;

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

setcolor(RED);

for(i=0;i<3;i++)

{

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

}

line( cl[3].x, cl[3].y, cl[0].x, cl[0].y);

setcolor(YELLOW);

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

{

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

}

getch();

suthodg(cl,4,dup\_sub,ns);

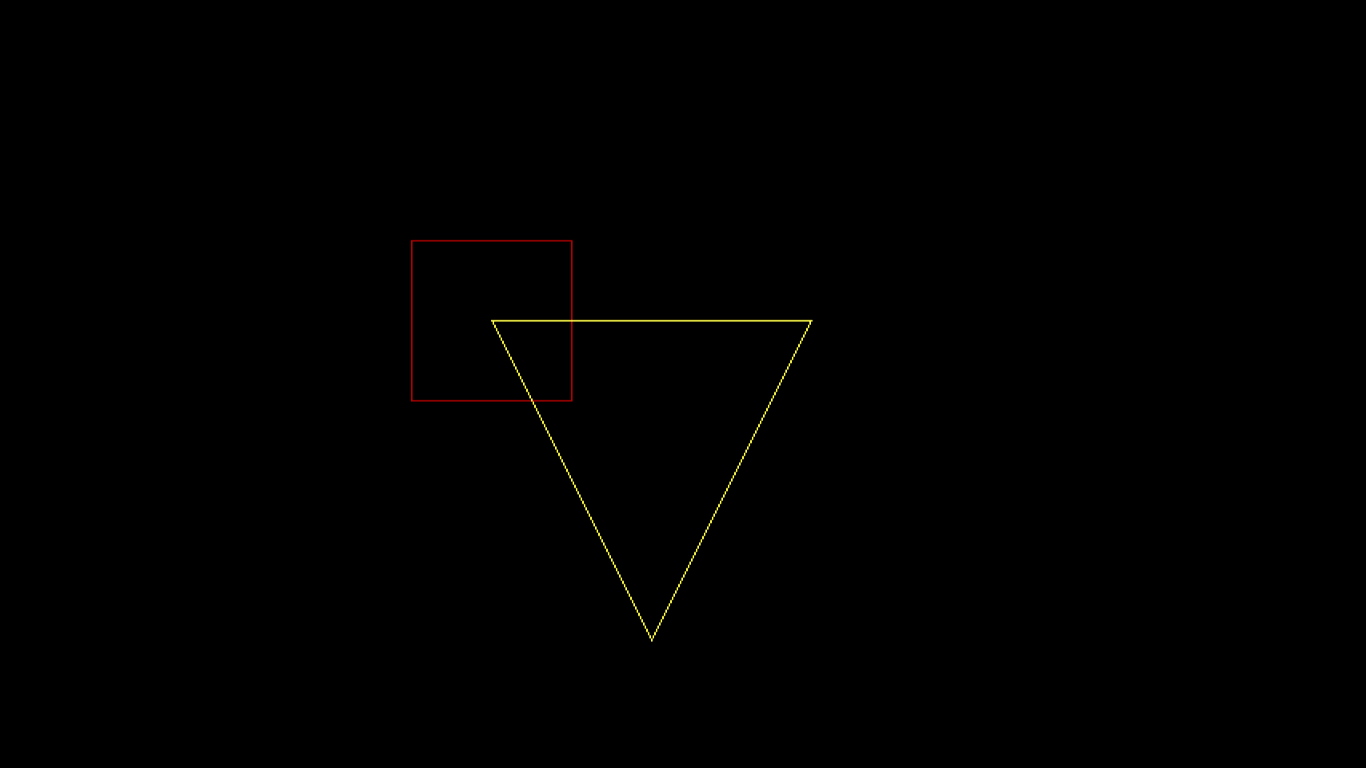
getch();

closegraph();

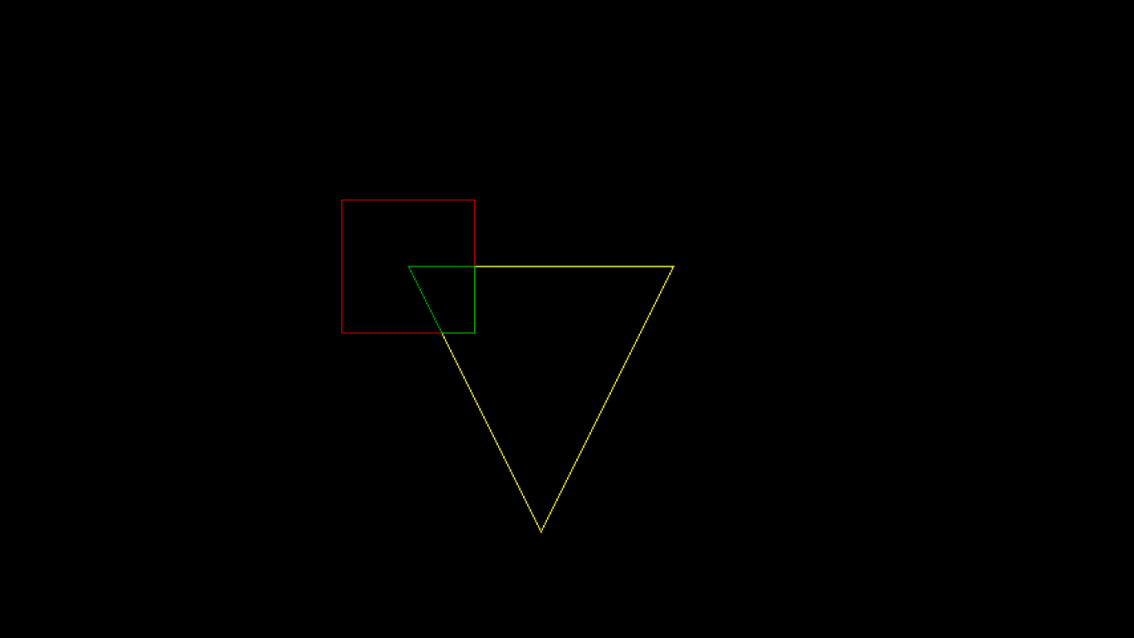
}

OUTPUT:

BEFORE CLIPPING



AFTER CLIPPING



PROGRAM – 16

Q – Draw a moving car.

CODE:

#include<iostream.h>

#include<graphics.h>

#include<conio.h>

#include<dos.h>

void main()

{ int gd=DETECT,gmode;

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

for(int i=0;i<600;i++)

{

/\*\*\*\*ROAD\*\*\*\*\*/

line(0,385,650,385);

/\*\*\*\*CAR BODY\*\*\*\*/

line(50+i,365,90+i,365);

arc(110+i,365,0,180,20);

line(130+i,365,220+i,365);

arc(240+i,365,0,180,20);

line(260+i,365,300+i,365);

line(300+i,365,300+i,330);

line(300+i,330,260+i,310);

line(260+i,310,230+i,280);

line(230+i,280,110+i,280);

line(110+i,280,70+i,320);

line(70+i,320,50+i,320);

line(50+i,320,50+i,365);

/\*\*\*\*CAR WHEELS\*\*\*\*/

circle(110+i,365,17);

circle(240+i,365,17);

/\*\*\*\*CAR WINDOWS\*\*\*\*/

line(250+i,310,225+i,285);

line(225+i,285,178+i,285);

line(178+i,285,178+i,310);

line(178+i,310,250+i,310);

line(173+i,285,115+i,285);

line(173+i,285,173+i,310);

line(173+i,310,90+i,310);

line(90+i,310,115+i,285);

delay(5);

cleardevice();

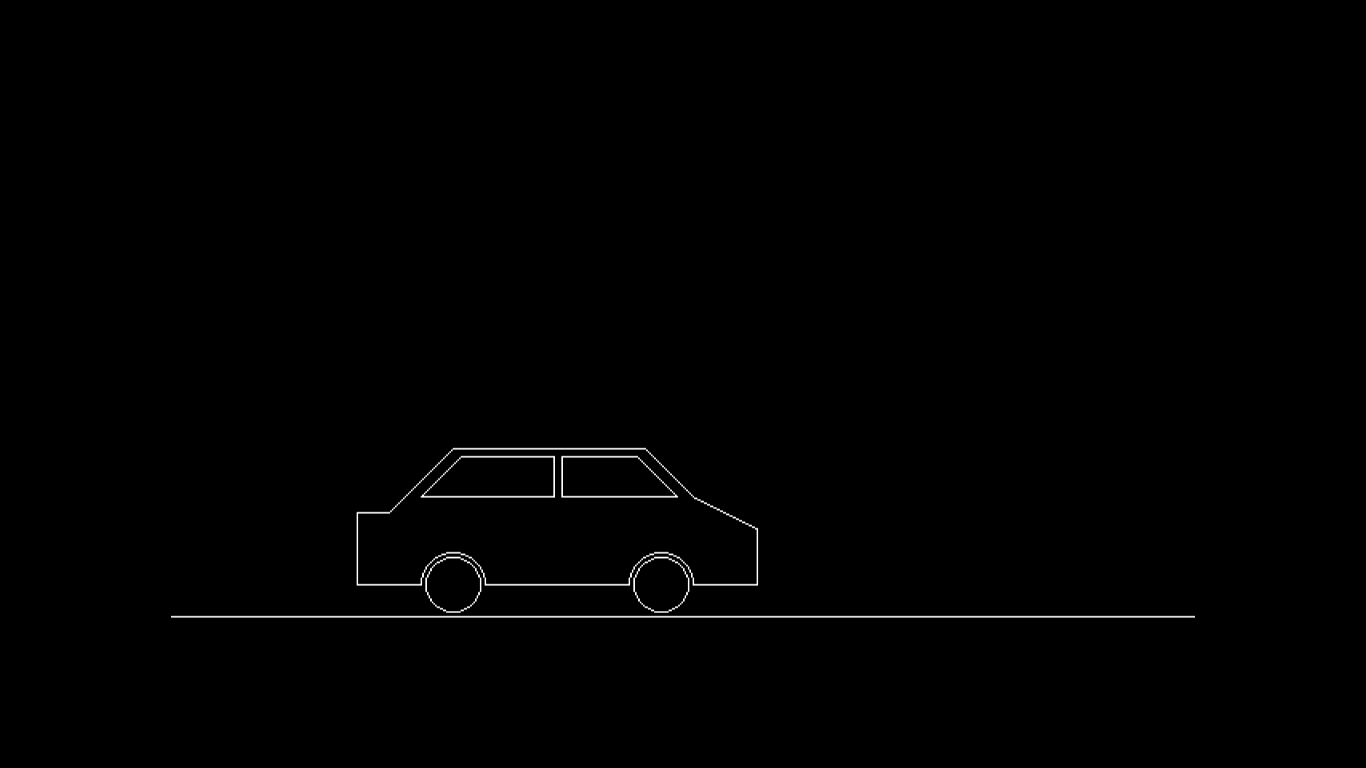
}

getch();

closegraph();

}

OUTPUT:



PROGRAM – 13

Q – Draw a hut with two windows.

CODE:

#include<iostream.h>

#include<graphics.h>

#include<conio.h>

void main()

{ clrscr();

int gd=DETECT,gmode;

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

setbkcolor(BLUE);

//creating front of hut

rectangle(150,250,250,400);

rectangle(250,250,450,400);

//creating triangular part of hut

line(150,250,200,200);

line(200,200,250,250);

line(250,250,150,250);

//creating roof of house

line(200,200,400,200);

line(400,200,450,250);

line(450,250,250,250);

line(250,250,200,200);

//creating door

rectangle(180,330,220,400);

//creating windows

rectangle(290,300,330,330);

rectangle(370,300,410,330);

//filling color in roof

setfillstyle(BKSLASH\_FILL,BROWN);

floodfill(250,240,WHITE);

//filling color in door

setfillstyle(SOLID\_FILL,YELLOW);

floodfill(200,350,WHITE);

//filling color in windows

setfillstyle(SOLID\_FILL,9);

floodfill(300,320,WHITE);

setfillstyle(SOLID\_FILL,9);

floodfill(400,320,WHITE);

//filling color in triangle

setfillstyle(SLASH\_FILL,BROWN);

floodfill(200,240,WHITE);

//filling color in front

setfillstyle(HATCH\_FILL,2);

floodfill(251,251,WHITE);

setfillstyle(CLOSE\_DOT\_FILL,RED);

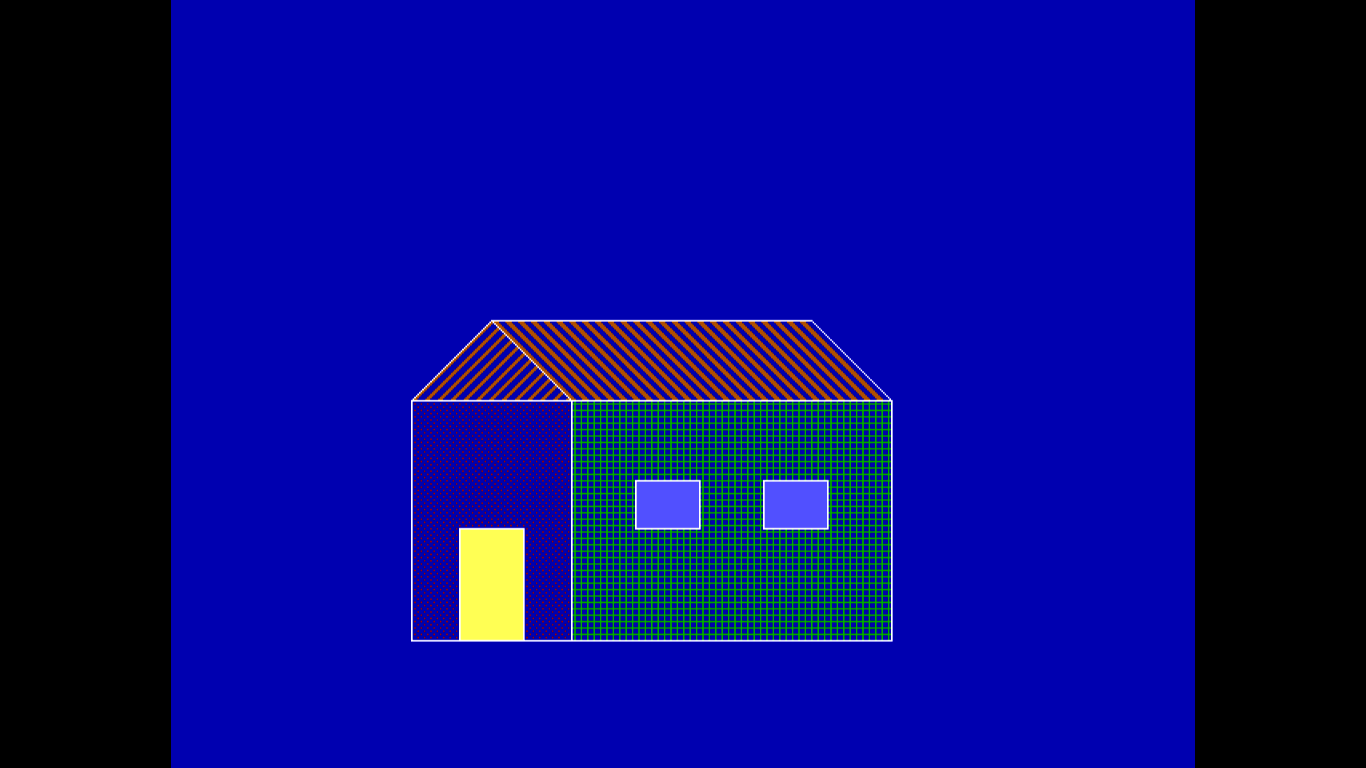
floodfill(151,251,WHITE);

getch();

closegraph();

}

OUTPUT:



PROGRAM – 4

Q – Write a program to rotate a triangle about origin.

CODE:

#include<iostream.h>

#include<conio.h>

#include <graphics.h>

#include <math.h>

#define ROUND(a) ((int) (a+0.5))

void lineDDA(int xa, int ya, int xb, int yb)

{ int dx=xb-xa, dy=yb-ya, steps, k;

float xinc, yinc, x=xa, y=ya;

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

steps=abs(dx);

else

steps=abs(dy);

xinc=dx/(float)steps;

yinc=dy/(float)steps;

putpixel(ROUND(x),ROUND(y),RED);

for(k=0; k<steps; k++)

{ x+=xinc;

y+=yinc;

putpixel(ROUND(x),ROUND(y),RED);

}

}

void main()

{

clrscr();

float angle;

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

cout<<"enter coordinates of vertex 1: ";

cin>>x1>>y1;

cout<<"enter coordinates of vertex 2: ";

cin>>x2>>y2;

cout<<"enter coordinates of vertex 3: ";

cin>>x3>>y3;

cout<<"Enter the angle of rotation: ";

cin>>angle;

angle=(3.14\*angle)/180;

float s=sin(angle);

float c=cos(angle);

float x1n,y1n,x2n,y2n,x3n,y3n;

x1n=(x1\*c)-(y1\*s);

y1n=(x1\*s)+(y1\*c);

x2n=(x2\*c)-(y2\*s);

y2n=(x2\*s)+(y2\*c);

x3n=(x3\*c)-(y3\*s);

y3n=(x3\*s)-(y3\*c);

int gd=DETECT,gm;

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

setbkcolor(YELLOW);

cout<<"ORIGINAL TRIANGLE:";

line(300,0,300,479);

line(0,240,639,240);

lineDDA(300+x1,240-y1,300+x2,240-y2);

lineDDA(300+x2,240-y2,300+x3,240-y3);

lineDDA(300+x3,240-y3,300+x1,240-y1);

getch();

clrscr();

cout<<"new triangle:";

line(300,0,300,479);

line(0,240,639,240);

lineDDA(300+x1n,240-y1n,300+x2n,240-y2n);

lineDDA(300+x2n,240-y2n,300+x3n,240-y3n);

lineDDA(300+x3n,240-y3n,300+x1n,240-y1n);

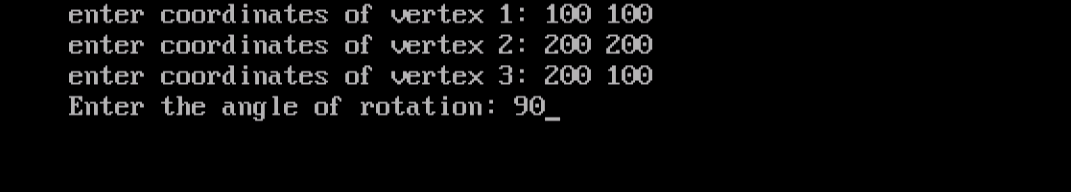
getch();

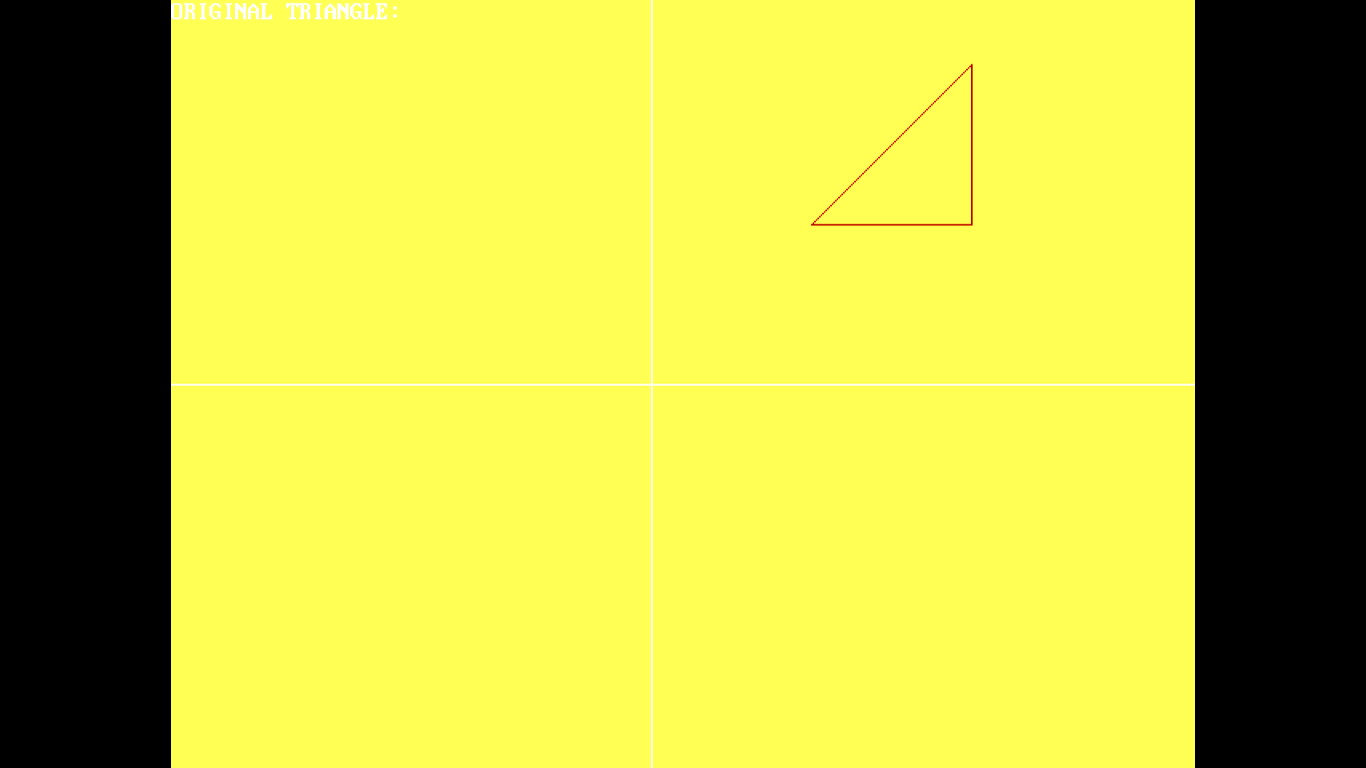
clrscr();

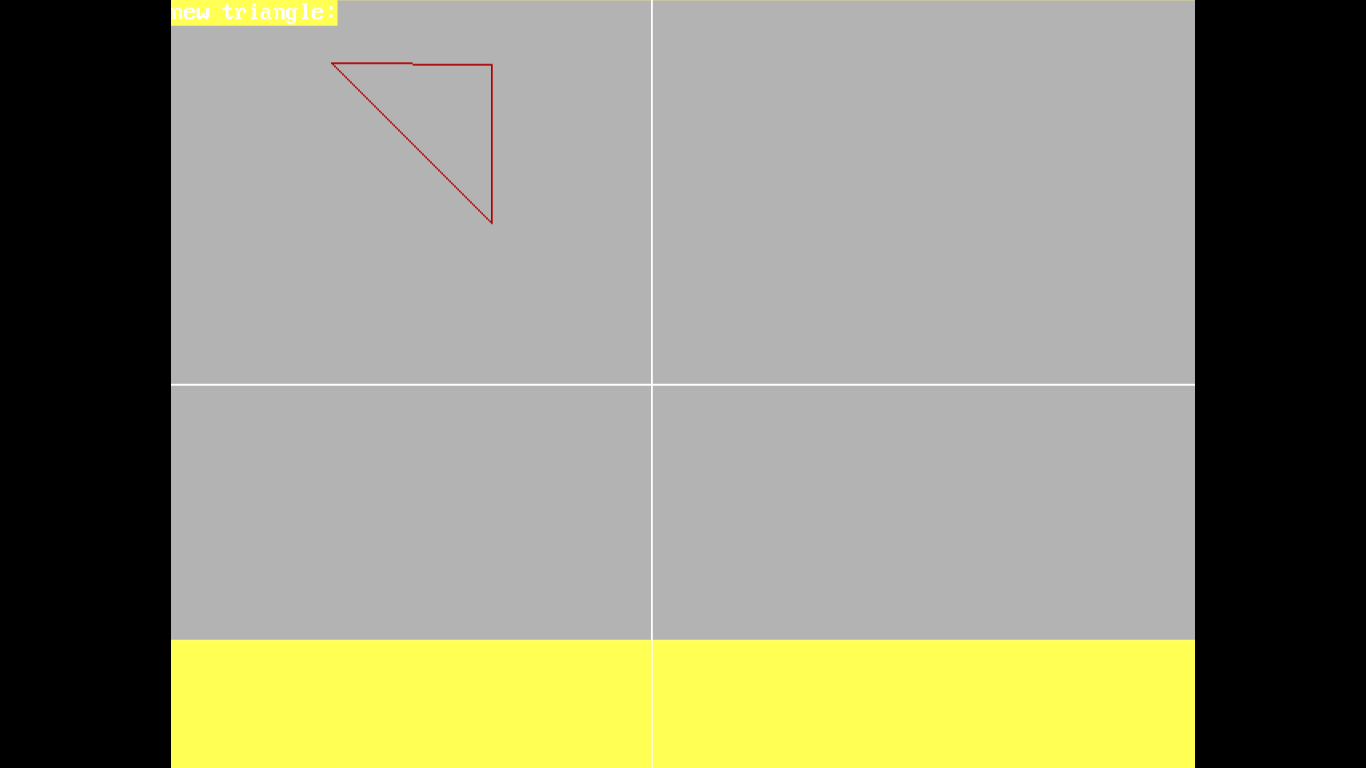
closegraph();

}

OUTPUT:







PROGRAM – 15

Q- Draw a bouncing ball.

CODE:

#include<iostream.h>

#include<conio.h>

#include<graphics.h>

#include<dos.h>

void main()

{

int gd=DETECT,gmode;

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

for(int k=0;k<=200;k++)

{ line(0,385,600,385);

circle(340,165+k,20);

delay(5);

cleardevice();

}

for(int i=0;i<=200;i++)

{ line(0,385,600,385);

circle(340,365-i,20);

delay(5);

cleardevice();

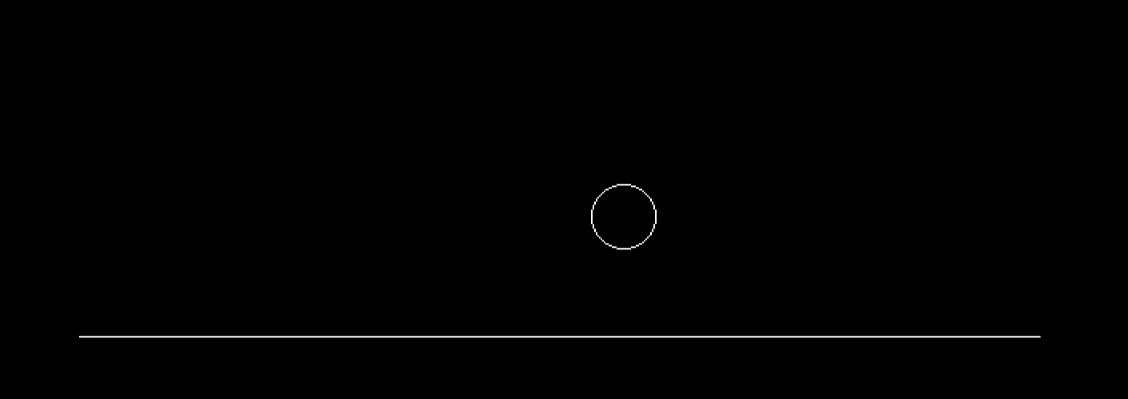
}

getch();

closegraph();

}

OUTPUT:



PROGRAM – 5

Q – Write a program to scale a triangle about origin.

CODE:

#include<iostream.h>

#include<conio.h>

#include <graphics.h>

#include <math.h>

#define ROUND(a) ((int) (a+0.5))

void lineDDA(int xa, int ya, int xb, int yb)

{ int dx=xb-xa, dy=yb-ya, steps, k;

float xinc, yinc, x=xa, y=ya;

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

steps=abs(dx);

else

steps=abs(dy);

xinc=dx/(float)steps;

yinc=dy/(float)steps;

putpixel(ROUND(x),ROUND(y),RED);

for(k=0; k<steps; k++)

{ x+=xinc;

y+=yinc;

putpixel(ROUND(x),ROUND(y),RED);

}

}

void main()

{

clrscr();

float sx,sy;

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

cout<<"enter coordinates of vertex 1: ";

cin>>x1>>y1;

cout<<"enter coordinates of vertex 2: ";

cin>>x2>>y2;

cout<<"enter coordinates of vertex 3: ";

cin>>x3>>y3;

cout<<"Enter the values of sx and sy: ";

cin>>sx>>sy;

int gd=DETECT,gm;

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

setbkcolor(YELLOW);

cout<<"ORIGINAL TRIANGLE:";

line(300,0,300,479);

line(0,240,639,240);

lineDDA(300+x1,240-y1,300+x2,240-y2);

lineDDA(300+x2,240-y2,300+x3,240-y3);

lineDDA(300+x3,240-y3,300+x1,240-y1);

getch();

clrscr();

x1=x1\*sx;

y1=y1\*sy;

x2=x2\*sx;

y2=y2\*sy;

x3=x3\*sx;

y3=y3\*sy;

cout<<"scaled triangle:";

line(300,0,300,479);

line(0,240,639,240);

lineDDA(300+x1,240-y1,300+x2,240-y2);

lineDDA(300+x2,240-y2,300+x3,240-y3);

lineDDA(300+x3,240-y3,300+x1,240-y1);

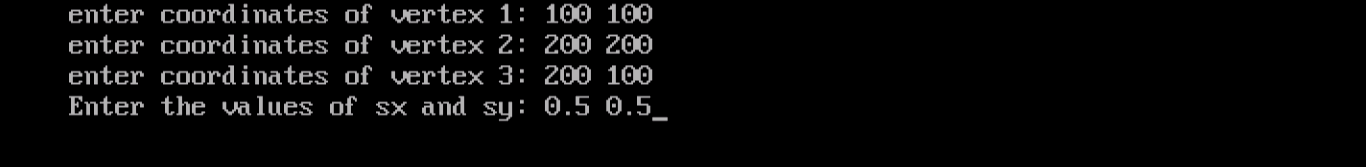
getch();

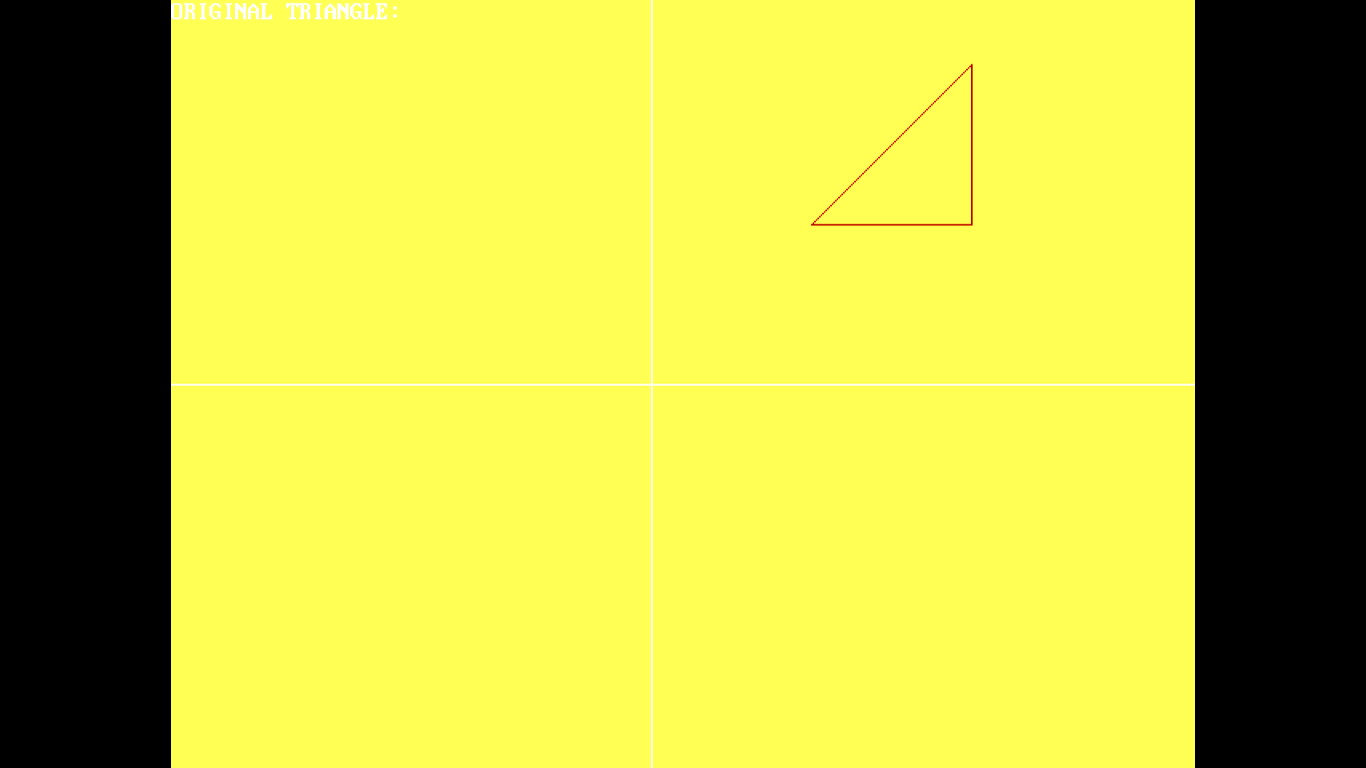
clrscr();

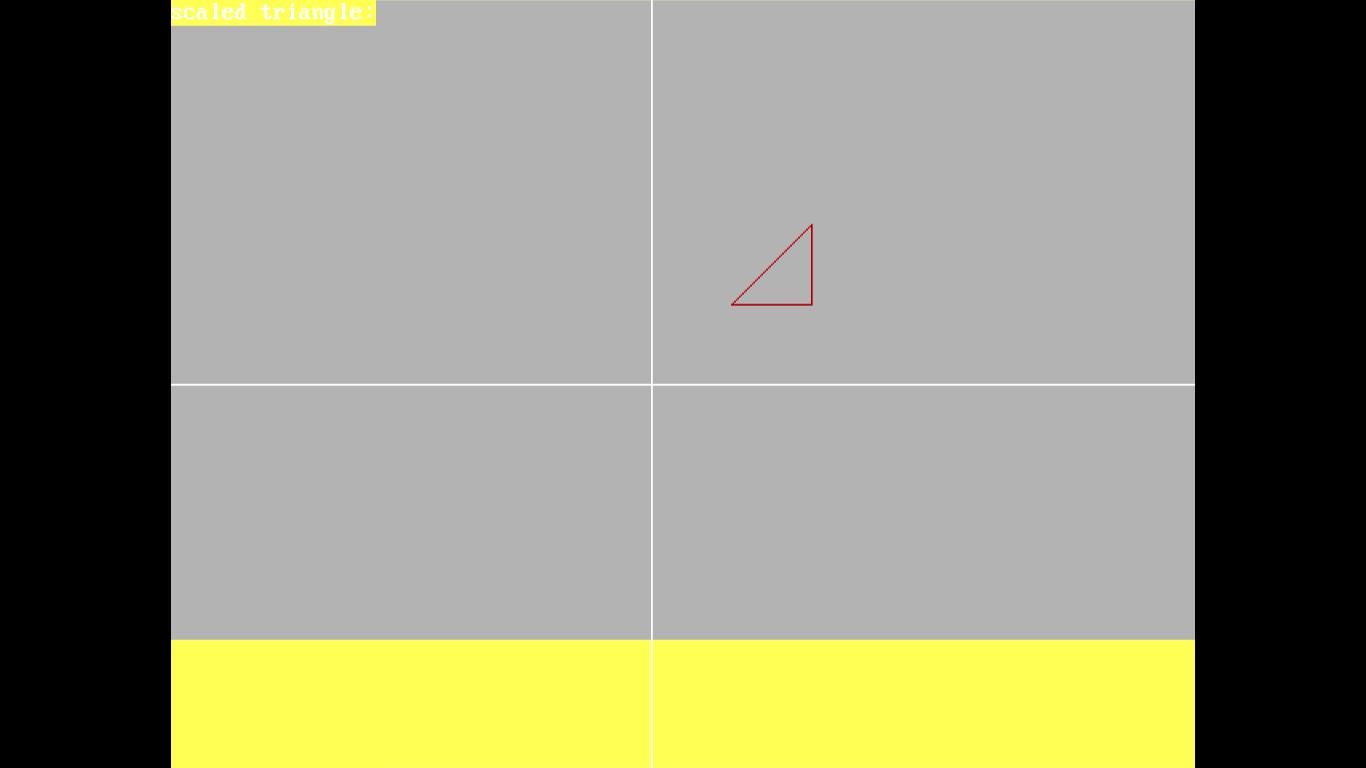
closegraph();

}

OUTPUT:







PROGRAM – 6

Q – Write a program to reflect a triangle.

CODE:

#include<iostream.h>

#include<graphics.h>

#include<conio.h>

void reflectx(int x1,int y1,int x2,int y2,int x3,int y3)

{

y1=-y1;

y2=-y2;

y3=-y3;

setcolor(BLUE);

line(300,0,300,479);

line(0,240,639,240);

line(300+x1,240-y1,300+x2,240-y2);

line(300+x3,240-y3,300+x2,240-y2);

line(300+x1,240-y1,300+x3,240-y3);

getch();

}

void reflecty(int x1,int y1,int x2,int y2,int x3,int y3)

{

x1=-x1;

x2=-x2;

x3=-x3;

setcolor(BLUE);

line(300,0,300,479);

line(0,240,639,240);

line(300+x1,240-y1,300+x2,240-y2);

line(300+x3,240-y3,300+x2,240-y2);

line(300+x1,240-y1,300+x3,240-y3);

getch();

}

void reflectxy(int x1,int y1,int x2,int y2,int x3,int y3)

{ int x1n,y1n,x2n,y2n,x3n,y3n;

x1n=-y1;

x2n=-y2;

x3n=-y3;

y1n=-x1;

y2n=-x2;

y3n=-x3;

setcolor(BLUE);

line(300,0,300,479);

line(0,240,639,240);

line(300+x1n,240-y1n,300+x2n,240-y2n);

line(300+x3n,240-y3n,300+x2n,240-y2n);

line(300+x1n,240-y1n,300+x3n,240-y3n);

getch();

}

void main()

{ int gd=DETECT,gmode;

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

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

cout<<"Enter the vertex 1 of triangle: ";

cin>>x1>>y1;

cout<<"Enter the vertex 2 of triangle: ";

cin>>x2>>y2;

cout<<"Enter the vertex 3 of triangle: ";

cin>>x3>>y3;

setcolor(RED);

line(300,0,300,479);

line(0,240,639,240);

line(300+x1,240-y1,300+x2,240-y2);

line(300+x3,240-y3,300+x2,240-y2);

line(300+x1,240-y1,300+x3,240-y3);

getch();

int ch;

cout<<"Enter your choice"<<endl;

cout<<"1. about x axis"<<endl;

cout<<"2. about y axis"<<endl;

cout<<"3. about y=x "<<endl;

cin>>ch;

switch(ch)

{

case 1:

reflectx(x1,y1,x2,y2,x3,y3);

break;

case 2:

reflecty(x1,y1,x2,y2,x3,y3);

break;

case 3:

reflectxy(x1,y1,x2,y2,x3,y3);

default:

cout<<"wrong input";

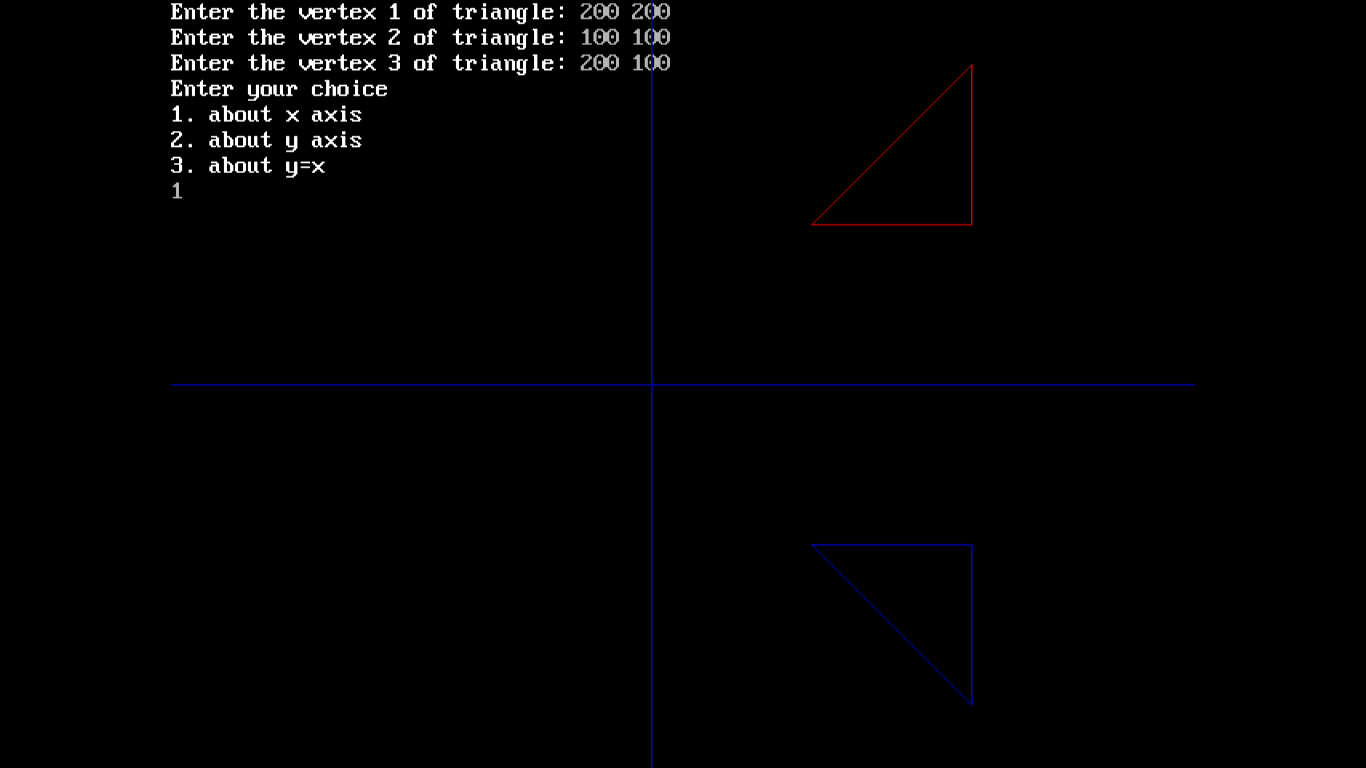
getch();

}

closegraph();

}

OUTPUT:



PROGRAM – 14

Q – Draw a bar chart having at least 3 bars.

CODE:

#include<iostream.h>

#include <graphics.h>

#include <conio.h>

int main() {

int gd = DETECT, gm;

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

settextstyle(8,0,2);

outtextxy(300,0,"BAR GRAPH");

setlinestyle(0,0,2);

/\* Draw X and Y Axis \*/ setcolor(5);

line(90,410,90,50);

line(90,410,590,410);

line(85,60,90,50);

line(95,60,90,50);

line(585,405,590,410);

line(585,415,590,410); setcolor(5);

outtextxy(65,60,"Y");

outtextxy(570,420,"X");

outtextxy(70,415,"O");

/\* Draw bars on screen \*/

setfillstyle(CLOSE\_DOT\_FILL, 11);

bar3d(150,80,200,410,15,2);

bar3d(225,100,275,410,15,2);

bar3d(300,120,350,410,15,2);

bar3d(375,170,425,410,15,2);

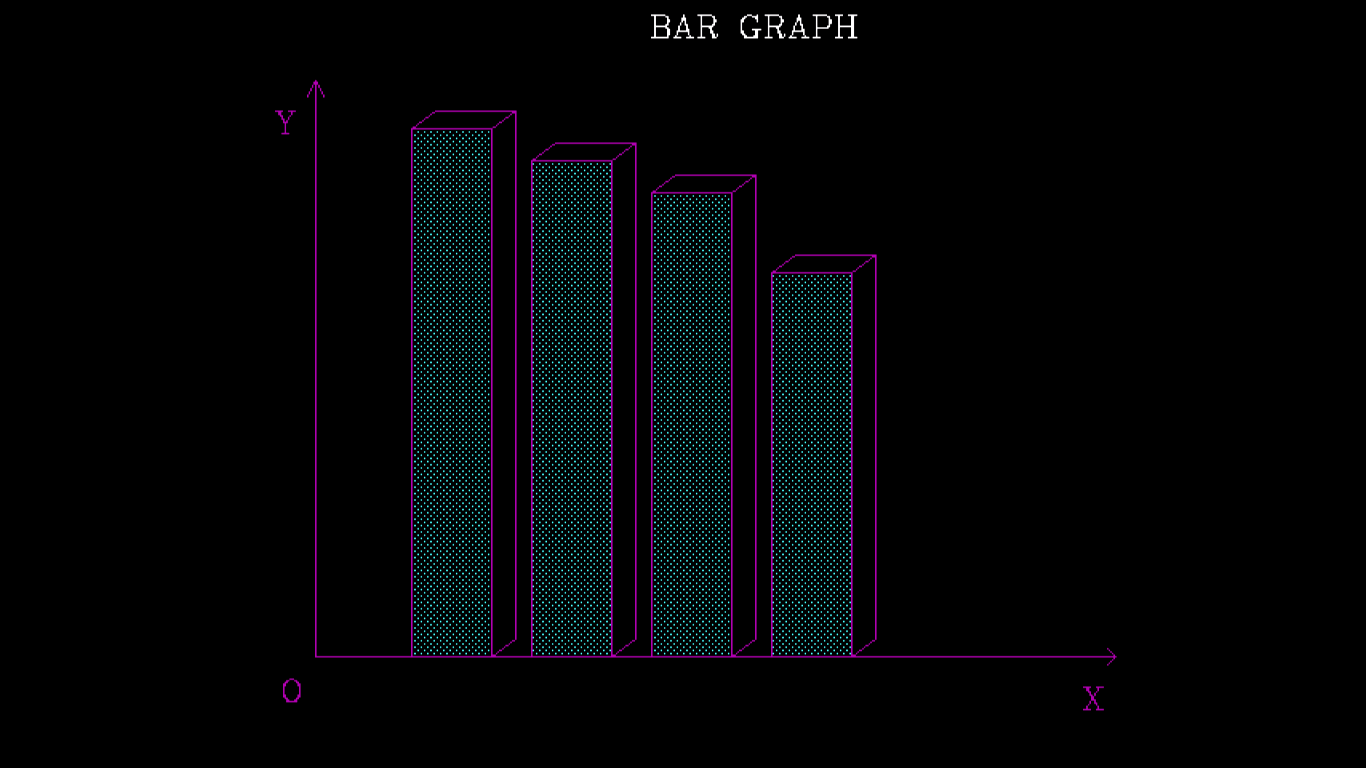
getch();

closegraph();

return 0;

}

OUTPUT:



PROGRAM – 12

Q – Draw a moon and three stars.

Code:

#include<iostream.h>

#include<conio.h>

#include<graphics.h>

#include<math.h>

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

{

int x[5],y[5],i;

float c=cos(72\*3.14/180);

float s=sin(72\*3.14/180);

x[0]=0; y[0]=r;

for(i=1;i<5;i++)

{ x[i]=(int)(x[i-1]\*c)-(y[i-1]\*s);

y[i]=(int)(x[i-1]\*s)+(y[i-1]\*c);

}

line(xc+x[0],-y[0]+yc,xc+x[2],-y[2]+yc);

line(xc+x[0],yc-y[0],xc+x[3],yc-y[3]);

line(xc+x[1],yc-y[1],xc+x[3],yc-y[3]);

line(xc+x[1],yc-y[1],xc+x[4],yc-y[4]);

line(xc+x[2],yc-y[2],xc+x[4],yc-y[4]);

}

void main()

{

int gd=DETECT,gmode;

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

setfillstyle(SOLID\_FILL,WHITE);

circle(200,100,50);

floodfill(200,100,WHITE);

star(500,200,10);

star(450,250,15);

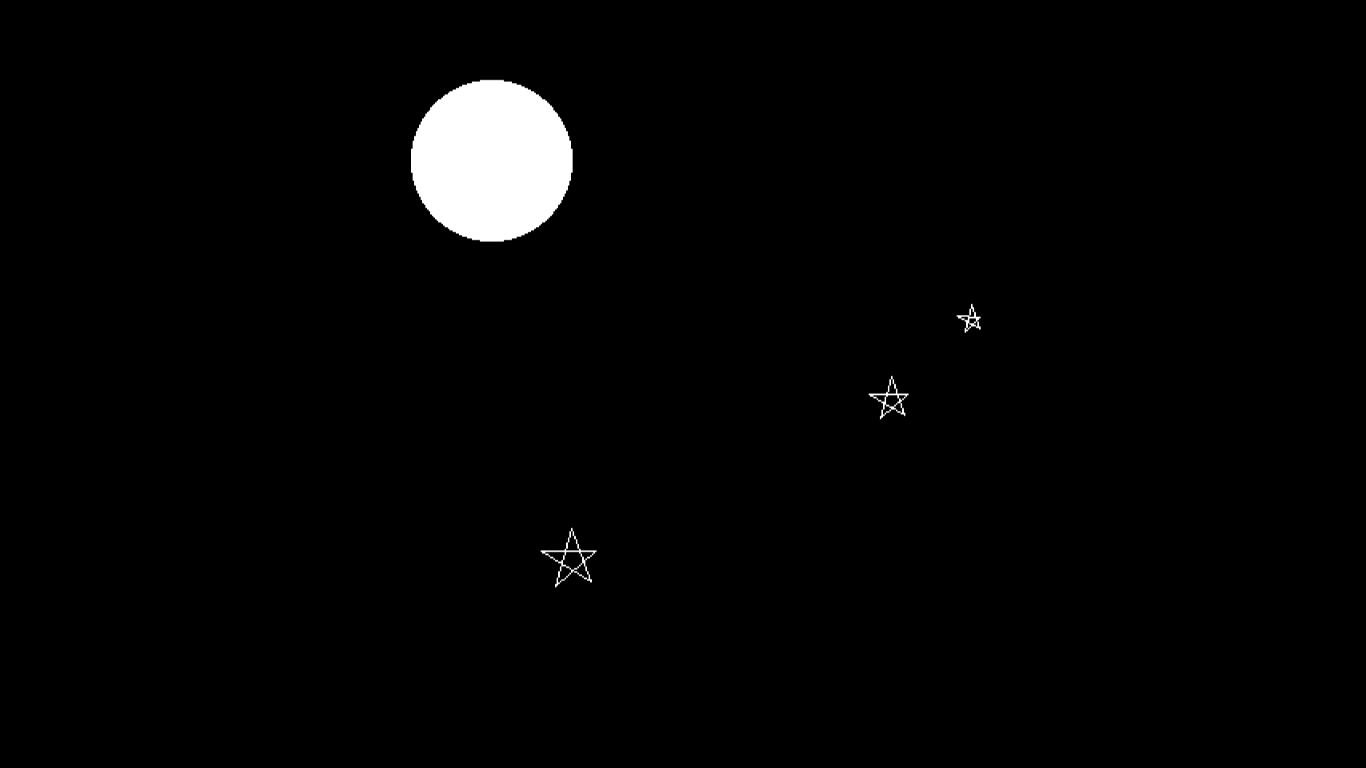
star(250,350,20);

getch();

closegraph();

}

OUTPUT:



PROGRAM - 18

Q – Draw a Laptop.

CODE:

#include<iostream.h>

#include<conio.h>

#include<graphics.h>

void main()

{

int gd=DETECT,gmode;

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

//drawing screen of laptop

line(250,200,350,200);

line(350,200,350,270);

line(350,270,250,270);

line(250,200,250,270);

line(240,190,360,190);

line(360,190,360,280);

line(240,280,360,280);

line(240,280,240,190);

//drawing keybord

line(240,280,275,315);

line(275,315,395,315);

line(395,315,360,280);

rectangle(275,315,395,320);

line(240,280,240,285);

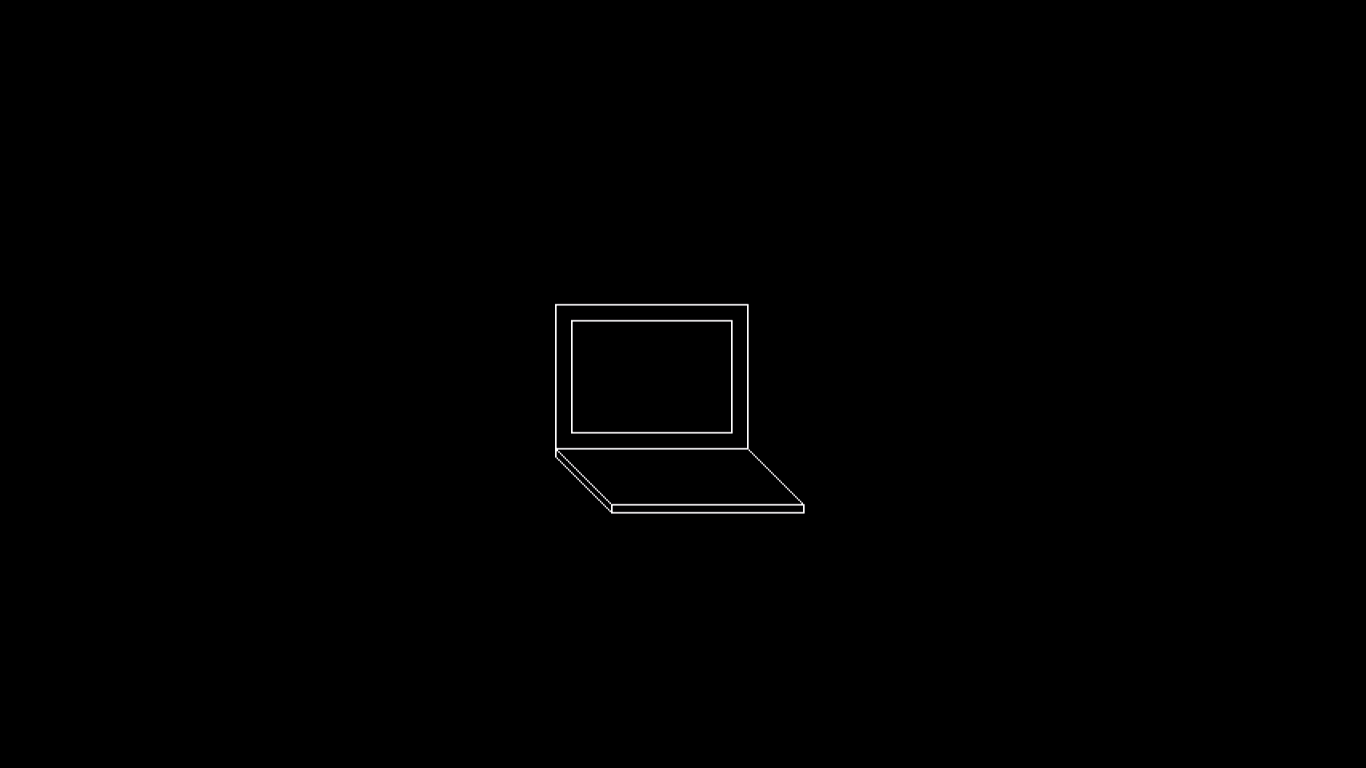
line(240,285,275,320);

getch();

closegraph();

}

OUTPUT:



PROGRAM – 17

Q – Write a program for displaying 3D objects as 2D display using perspective transformation. Also write a program for 3D transformation.

CODE:

#include <iostream>

#include <vector>

#include <stdio.h>

#include <stdlib.h>

#include <graphics.h>

#include "shapes.h"

#include <math.h>

#include <dos.h>

#define PI 3.14159

using namespace std;

int XX, YY;

struct point3D

{

float x, y, z;

point3D()

{

x=y=z=0;

}

point3D(float a, float b, float c)

{

x=a; y=b; z=c;

}

};

typedef vector<point3D> VERTICES;

typedef vector< vector<int> > EDGES;

struct POLY3D

{

VERTICES V;

EDGES E;

point3D centroid;

POLY3D(int N)

{

E = EDGES(N+1, vector<int>() );

}

};

void mulmat3D(float A[4][4], float B[4][4], float C[4][4])

{

for(int i=0; i<4; ++i)

for(int j=0; j<4; ++j)

{

C[i][j] = 0;

for(int k=0; k<4; ++k)

C[i][j] += A[i][k]\*B[k][j];

}

}

point3D mulpoint3D(point3D p, float C[4][4])

{

point3D res;

float h = 0.0;

res.x = C[0][0]\*p.x + C[0][1]\*p.y + C[0][2]\*p.z + C[0][3];

res.y = C[1][0]\*p.x + C[1][1]\*p.y + C[1][2]\*p.z + C[1][3];

res.z = C[2][0]\*p.x + C[2][1]\*p.y + C[2][2]\*p.z + C[2][3];

h = C[3][0]\*p.x + C[3][1]\*p.y + C[3][2]\*p.z + C[3][3];

res.x /= h;

res.y /= h;

res.z /= h;

return res;

}

void trans3D(float dx, float dy, float dz, float C[4][4])

{

for(int i=0; i<4; ++i)

for(int j=0; j<4; ++j)

{

if(i==j)

C[i][j] = 1;

else

C[i][j] = 0;

}

C[0][3] = dx;

C[1][3] = dy;

C[2][3] = dz;

}

void rot3D(float theta, float C[4][4], int axis) // z=0, x=1, y=2

{

theta = theta\*PI/180.0;

for(int i=0; i<4; ++i)

for(int j=0; j<4; ++j)

{

if(i==j)

C[i][j] = 1;

else

C[i][j] = 0;

}

C[axis][axis] = C[(axis+1)%3][(axis+1)%3] = cos(theta);

C[(axis+1)%3][axis] = sin(theta);

C[axis][(axis+1)%3] = -sin(theta);

}

void scale3D(float sx, float sy, float sz, float C[4][4])

{

for(int i=0; i<4; ++i)

for(int j=0; j<4; ++j)

{

if(i==j)

C[i][j] = 1;

else

C[i][j] = 0;

}

C[0][0] = sx;

C[1][1] = sy;

C[2][2] = sz;

}

void plotline3D(point3D p1, point3D p2)

{

LINE(p1.x+XX, YY-p1.y, p2.x+XX, YY-p2.y);

//LINE(p1.x, p1.y, p2.x, p2.y);

}

void plotpoly3D(POLY3D &POLY)

{

for(int i=0; i<POLY.E.size(); ++i)

for(int j=0; j<POLY.E[i].size(); ++j)

plotline3D(POLY.V[i], POLY.V[POLY.E[i][j]]);

}

void drawaxes()

{

setcolor(WHITE);

LINE(getmaxx()/2, 0, getmaxx()/2, getmaxy());

LINE(0, getmaxy()/2, getmaxx(), getmaxy()/2);

}

void changePOLY(POLY3D &POLY, float C[4][4])

{

for(int i=0; i<POLY.V.size(); ++i)

POLY.V[i] = mulpoint3D(POLY.V[i], C);

}

int main()

{

initwindow(1300, 700, "GRAPHICS");

XX = getmaxx()/2, YY = getmaxy()/2;

cout<<"No. of vertices and edges of polygon : ";

int N, M;

cin>>N>>M;

POLY3D POLY(N);

cout<<"Vertices\n";

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

{

int x, y, z;

cin>>x>>y>>z;

POLY.V.push\_back(point3D(x, y, z));

POLY.centroid.x += x;

POLY.centroid.y += y;

POLY.centroid.z += z;

}

POLY.centroid.x /= N;

POLY.centroid.y /= N;

POLY.centroid.z /= N;

cout<<"Edges\n";

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

{

int p1, p2;

cin>>p1>>p2;

POLY.E[p1].push\_back(p2);

}

setcolor(YELLOW);

plotpoly3D(POLY);

int ch = 0;

float C[4][4], A[4][4], B[4][4];

while(ch!=-1)

{

ch = getch();

switch(ch)

{

case 'z' : trans3D(-POLY.centroid.x, -POLY.centroid.y, -POLY.centroid.z, C);

rot3D(10, A, 0);

mulmat3D(A, C, B);

trans3D(POLY.centroid.x, POLY.centroid.y, POLY.centroid.z, A);

mulmat3D(A, B, C);

changePOLY(POLY, C);

break;

case 'x' : trans3D(-POLY.centroid.x, -POLY.centroid.y, -POLY.centroid.z, C);

rot3D(-10, A, 0);

mulmat3D(A, C, B);

trans3D(POLY.centroid.x, POLY.centroid.y, POLY.centroid.z, A);

mulmat3D(A, B, C);

changePOLY(POLY, C);

break;

case 'w' : trans3D(-POLY.centroid.x, -POLY.centroid.y, -POLY.centroid.z, C);

rot3D(10, A, 1);

mulmat3D(A, C, B);

trans3D(POLY.centroid.x, POLY.centroid.y, POLY.centroid.z, A);

mulmat3D(A, B, C);

changePOLY(POLY, C);

break;

case 's' : trans3D(-POLY.centroid.x, -POLY.centroid.y, -POLY.centroid.z, C);

rot3D(-10, A, 1);

mulmat3D(A, C, B);

trans3D(POLY.centroid.x, POLY.centroid.y, POLY.centroid.z, A);

mulmat3D(A, B, C);

changePOLY(POLY, C);

break;

case 'a' : trans3D(-POLY.centroid.x, -POLY.centroid.y, -POLY.centroid.z, C);

rot3D(10, A, 2);

mulmat3D(A, C, B);

trans3D(POLY.centroid.x, POLY.centroid.y, POLY.centroid.z, A);

mulmat3D(A, B, C);

changePOLY(POLY, C);

break;

case 'd' : trans3D(-POLY.centroid.x, -POLY.centroid.y, -POLY.centroid.z, C);

rot3D(-10, A, 2);

mulmat3D(A, C, B);

trans3D(POLY.centroid.x, POLY.centroid.y, POLY.centroid.z, A);

mulmat3D(A, B, C);

changePOLY(POLY, C);

break;

case '+' : trans3D(-POLY.centroid.x, -POLY.centroid.y, -POLY.centroid.z, C);

scale3D(1.1, 1.1, 1.1, A);

mulmat3D(A, C, B);

trans3D(POLY.centroid.x, POLY.centroid.y, POLY.centroid.z, A);

mulmat3D(A, B, C);

changePOLY(POLY, C);

break;

case '-' : trans3D(-POLY.centroid.x, -POLY.centroid.y, -POLY.centroid.z, C);

scale3D(0.9, 0.9, 0.9, A);

mulmat3D(A, C, B);

trans3D(POLY.centroid.x, POLY.centroid.y, POLY.centroid.z, A);

mulmat3D(A, B, C);

changePOLY(POLY, C);

break;

default: continue;

}

cleardevice();

setcolor(YELLOW);

setlinestyle(SOLID\_LINE, 1, 3);

plotpoly3D(POLY);

}

return 0;

}

OUTPUT:

