***Experiment No.1: Draw a line using Bresenham’s Line Algorithm.***

**Source Code**

#include<iostream.h>

#include<graphics.h>

Using namespace std;

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;

    }

}

 int main()

{

    int gdriver=DETECT, gmode, error, x0, y0, x1, y1;

    initgraph(&gdriver, &gmode, "c:\\turboc3\\bgi");

  cout<<"Enter co-ordinates of first point: ";

    cin>>x0>>y0;

   cout<<"Enter co-ordinates of second point: ";

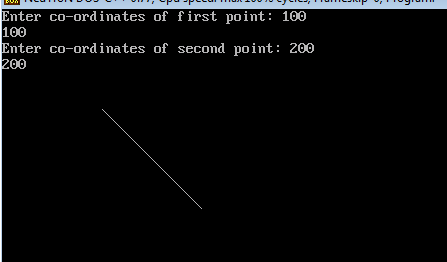
    cin>>x1>>y1;

    drawline(x0, y0, x1, y1);

  return 0;

}

**Output:**

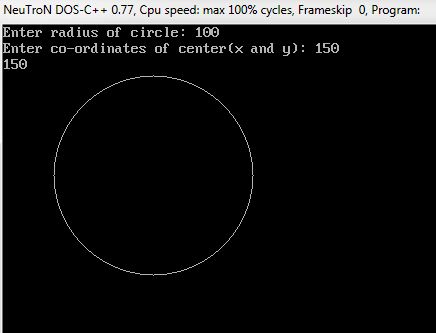


***Experiment No. 2: Draw a Circle using Bresenham Circle Algorithm.***

**Source Code**

# include<stdio.h>  
# include<conio.h>  
# include<graphics.h>  
# include<math.h>  
int main()  
{  
int gd=DETECT,gm;  
int r,x,y,p,xc=320,yc=240;  
initgraph(&gd,&gm,"C:\\TC\\BGI");  
cleardevice();  
printf("Enter the radius ");  
scanf("%d",&r);  
x=0;  
y=r;  
putpixel(xc+x,yc-y,1);  
p=3-(2\*r);  
for(x=0;x<=y;x++)  
{  
if (p<0)  
{  
y=y;  
p=(p+(4\*x)+6);  
}  
else  
{  
y=y-1;  
p=p+((4\*(x-y)+10));  
}  
putpixel(xc+x,yc-y,1);  
putpixel(xc-x,yc-y,2);  
putpixel(xc+x,yc+y,3);  
putpixel(xc-x,yc+y,4);  
putpixel(xc+y,yc-x,5);putpixel(xc-y,yc-x,6);  
putpixel(xc+y,yc+x,7);  
putpixel(xc-y,yc+x,8);  
}  
getch();  
closegraph();  
}

**Output:**

****

***Experiment No.3: Draw a Circle using Midpoint algorithm.***

**Source Code**

#include<bits/stdc++.h>

#include<conio.h>

#include<graphics.h>

using namespace std;

void pixel(int xc,int yc,int x,int y);

int main()

{

int gd=DETECT,gm,xc,yc,r,x,y,Pk;

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

printf("\t\t Mid-Point algorithm of circle\n");

printf("Enter the value of X:\t");

scanf("%d",&xc);

printf("Enter the value of Y:\t");

scanf("%d",&yc);

printf("Enter the Radius of circle\t");

scanf("%d",&r);

x=0;

y=r;

Pk=1-r;

pixel(xc,yc,x,y);

while(x<y)

{

if(Pk<0)

{

x=x+1;

Pk=Pk+(2\*x)+1;

}

else

{

x=x+1;

y=y-1;

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

}

pixel(xc,yc,x,y);

}

getch();

closegraph();

}

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

{

putpixel(xc+x,yc+y,7);

putpixel(xc+y,yc+x,7);

putpixel(xc-y,yc+x,7);

putpixel(xc-x,yc+y,7);

putpixel(xc-x,yc-y,7);

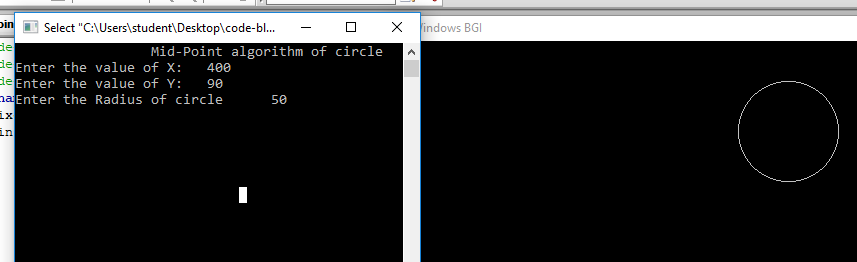
putpixel(xc-y,yc-x,7);

putpixel(xc+y,yc-x,7);

putpixel(xc+x,yc-y,7);

}

**Output:**

******

***Experiment No.4: Ellipse drawing.***

**Source Code**

#include<bits/stdc++.h>

#include<graphics.h>

#include<conio.h>

using namespace std;

int main(){

int gd = DETECT,gm;

int x ,y;

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

x = getmaxx()/2;

y = getmaxy()/2;

outtextxy(x-100, 50, "ELLIPSE drawing");

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

getch();

closegraph();

return 0;

}

**Output:**



**Experiment No.5: Moving Car**

**Source Code:**

#include <stdio.h>

#include <graphics.h>

#include <conio.h>

#include <dos.h>

int main() {

    int gd = DETECT, gm;

    int i, maxx, midy;

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

    maxx = getmaxx();

    midy = getmaxy()/2;

    for (i=0; i < maxx-150; i=i+5) {

        cleardevice();

        setcolor(WHITE);

        line(0, midy + 37, maxx, midy + 37);

        setcolor(YELLOW);

        setfillstyle(SOLID\_FILL, RED);

        line(i, midy + 23, i, midy);

        line(i, midy, 40 + i, midy - 20);

        line(40 + i, midy - 20, 80 + i, midy - 20);

        line(80 + i, midy - 20, 100 + i, midy);

        line(100 + i, midy, 120 + i, midy);

        line(120 + i, midy, 120 + i, midy + 23);

        line(0 + i, midy + 23, 18 + i, midy + 23);

        arc(30 + i, midy + 23, 0, 180, 12);

        line(42 + i, midy + 23, 78 + i, midy + 23);

arc(90 + i, midy + 23, 0, 180, 12);

        line(102 + i, midy + 23, 120 + i, midy + 23);

        line(28 + i, midy, 43 + i, midy - 15);

        line(43 + i, midy - 15, 57 + i, midy - 15);

        line(57 + i, midy - 15, 57 + i, midy);

        line(57 + i, midy, 28 + i, midy);

        line(62 + i, midy - 15, 77 + i, midy - 15);

        line(77 + i, midy - 15, 92 + i, midy);

        line(92 + i, midy, 62 + i, midy);

        line(62 + i, midy, 62 + i, midy - 15);

        floodfill(5 + i, midy + 22, YELLOW);

        setcolor(BLUE);

        setfillstyle(SOLID\_FILL, DARKGRAY);

        /\*  Draw Wheels \*/

        circle(30 + i, midy + 25, 9);

        circle(90 + i, midy + 25, 9);

        floodfill(30 + i, midy + 25, BLUE);

        floodfill(90 + i, midy + 25, BLUE);

        delay(100);

    }

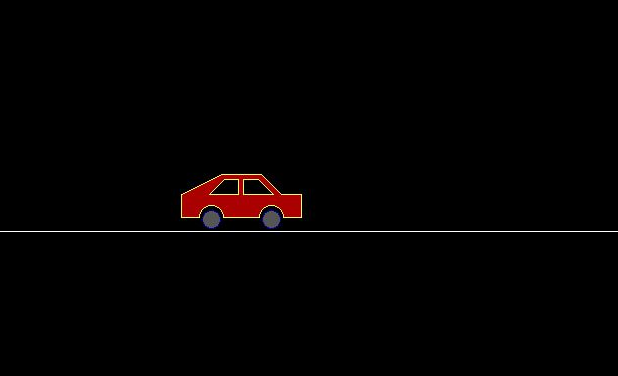
    getch();

    closegraph();

    return 0;

}

**Output:**



**Experiment No. 6: Water fall on a bucket.**

**Source Code:**

#include"stdio.h"

#include"graphics.h"

#include"math.h"

int main()

{

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

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

ellipse(300,200,0,360,50,25);

ellipse(300,300,0,360,50,25);

line(250,200,250,300);

line(350,200,350,300);

ellipse(300,100,180,360,5,2);

line(295,100,295,80);

line(305,100,305,86);

arc(300,80,90,180,5);

putpixel(306,85,15);

putpixel(307,84,15);

line(308,84,630,84);

line(300,75,303,75);

line(314,75,630,75);

putpixel(304,74,15);

putpixel(305,73,15);

line(306,72,306,65);

line(311,72,311,65);

putpixel(312,73,15);

putpixel(313,74,15);

pieslice(309,62,0,360,5);

setfillstyle(SOLID\_FILL,BLUE);

setcolor(BLUE);

for(i=0;i<7;i++)

{

line(297+i,103,297+i,300);

}

for(i=1;i<100;i++)

{

setcolor(BLUE);

ellipse(300,300-i,180,360,4,2);

delay(30);

fillellipse(300,300-i,49,25);

setcolor(1);

line(297,275-i,303,275-i);

setcolor(15);

ellipse(300,200,180,360,50,25);

delay(50);

}

ellipse(300,200,0,360,50,25);

setcolor(0);

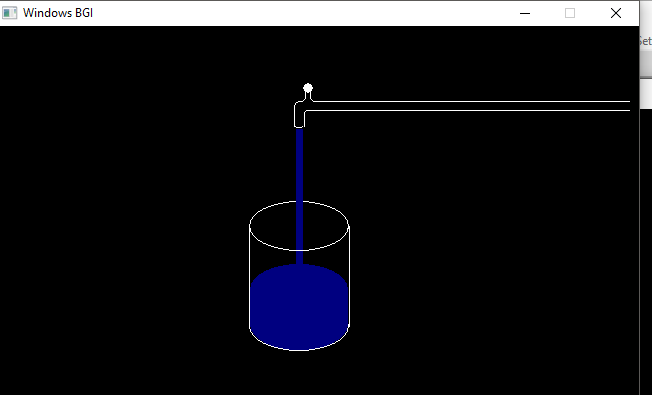
for(i=0;i<7;i++)

line(297+i,103,297+i,174);

getch();

}

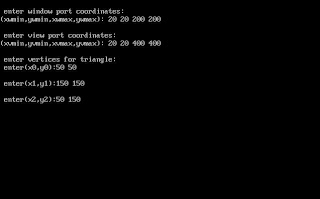
**Output:**

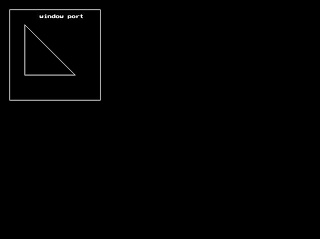
****

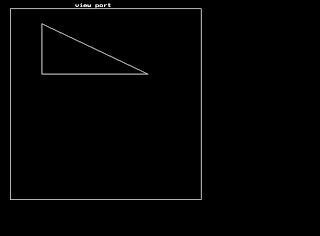
***Experiment No.9: Window to Viewport Mapping***

**Source Code:**

#include"stdio.h"  
#include"conio.h"  
#include"graphics.h"  
#include"stdlib.h"  
void main()  
{  
    float xwmin,xwmax,ywmax,ywmin;  
    float xvmin,xvmax,yvmax,yvmin;  
    float x[10],y[10],yv,xv,sx,sy;  
    int gd=DETECT,gm,i;  
    clrscr();  
    printf("\n enter window port coordinates:\n(xwmin,ywmin,xwmax,ywmax): ");  
    scanf("%f%f%f%f",&xwmin,&ywmin,&xwmax,&ywmax);  
    printf("\n enter view port coordinates:\n(xvmin,yvmin,xvmax,yvmax): ");  
    scanf("%f%f%f%f",&xvmin,&yvmin,&xvmax,&yvmax);  
    printf("\n enter vertices for triangle: ");  
    for(i=0;i < 3;i++)  
    {  
    printf("\n enter(x%d,y%d):",i,i);  
    scanf("%f%f",&x[i],&y[i]);  
    }  
    sx=((xvmax-xvmin)/(xwmax-xwmin));  
    sy=((yvmax-yvmin)/(ywmax-xwmin));  
  
    initgraph(&gd,&gm," ");  
    outtextxy(80,30,"window port");  
    rectangle(xwmin,ywmin,xwmax,ywmax);  
    for(i=0;i <2;i++)  
    {  
    line(x[i],y[i],x[i+1],y[i+1]);  
    }  
    line(x[2],y[2],x[0],y[0]);  
    getch();  
    cleardevice();  
    for(i=0;i <3;i++)  
    {  
    x[i]=xvmin+((x[i]-xwmin)\*sx);  
    y[i]=yvmin+((y[i]-ywmin)\*sy);  
    }  
    outtextxy(150,10,"view port");  
    rectangle(xvmin,yvmin,xvmax,yvmax);  
    for(i=0;i <2;i++)  
    {  
    line(x[i],y[i],x[i+1],y[i+1]);  
    }  
    line(x[2],y[2],x[0],y[0]);  
    getch();  
}  
  
OUTPUT:

[](http://3.bp.blogspot.com/_j82bvWjy4n4/S4ZNTMS2kKI/AAAAAAAAADk/EqXafr0-OA8/s1600-h/untitled.JPG)

[](http://3.bp.blogspot.com/_j82bvWjy4n4/S4ZNadl9vtI/AAAAAAAAADs/rlW8buHjP50/s1600-h/untitled1.JPG)

[](http://2.bp.blogspot.com/_j82bvWjy4n4/S4ZNbXnUHJI/AAAAAAAAAD0/dfGMhOHEZxw/s1600-h/untitled2.JPG)

***Experiment No.10: Cohen Sutherland Line Clipping Algorithm***

**Source Code:**

#include<iostream.h>

#include<stdlib.h>

#include<math.h>

#include<graphics.h>

#include<dos.h>

typedef struct coordinate

{

    int x,y;

    char code[4];

}PT;

void drawwindow();

void drawline(PT p1,PT p2);

PT setcode(PT p);

int visibility(PT p1,PT p2);

PT resetendpt(PT p1,PT p2);

void main()

{

    int gd=DETECT,v,gm;

    PT p1,p2,p3,p4,ptemp;

    cout<<"\nEnter x1 and y1\n";

    cin>>p1.x>>p1.y;

    cout<<"\nEnter x2 and y2\n";

    cin>>p2.x>>p2.y;

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

    drawwindow();

    delay(500);

    drawline(p1,p2);

    delay(500);

    cleardevice();

    delay(500);

    p1=setcode(p1);

    p2=setcode(p2);

    v=visibility(p1,p2);

    delay(500);

    switch(v)

    {

    case 0: drawwindow();

            delay(500);

            drawline(p1,p2);

            break;

    case 1:    drawwindow();

            delay(500);

            break;

    case 2:    p3=resetendpt(p1,p2);

            p4=resetendpt(p2,p1);

            drawwindow();

            delay(500);

            drawline(p3,p4);

            break;

    }

    delay(5000);

    closegraph();

}

 void drawwindow()

{

    line(150,100,450,100);

    line(450,100,450,350);

    line(450,350,150,350);

    line(150,350,150,100);

}

 void drawline(PT p1,PT p2)

{

    line(p1.x,p1.y,p2.x,p2.y);

}

PT setcode(PT p)    //for setting the 4 bit code

{

    PT ptemp;

    if(p.y<100)

        ptemp.code[0]='1';    //Top

    else

        ptemp.code[0]='0';

    if(p.y>350)

        ptemp.code[1]='1';    //Bottom

    else

        ptemp.code[1]='0';

    if(p.x>450)

        ptemp.code[2]='1';    //Right

    else

        ptemp.code[2]='0';

    if(p.x<150)

        ptemp.code[3]='1';    //Left

    else

        ptemp.code[3]='0';

    ptemp.x=p.x;

    ptemp.y=p.y;

    return(ptemp);

}

int visibility(PT p1,PT p2)

{

    int i,flag=0;

    for(i=0;i<4;i++)

    {

        if((p1.code[i]!='0') || (p2.code[i]!='0'))

            flag=1;

    }

     if(flag==0)

        return(0);

    for(i=0;i<4;i++)

    {

        if((p1.code[i]==p2.code[i]) && (p1.code[i]=='1'))

            flag='0';

    }

     if(flag==0)

        return(1);

    return(2);

}

 PT resetendpt(PT p1,PT p2)

{

    PT temp;

    int x,y,i;

    float m,k;

    if(p1.code[3]=='1')

        x=150;

    if(p1.code[2]=='1')

        x=450;

    if((p1.code[3]=='1') || (p1.code[2]=='1'))

    {

        m=(float)(p2.y-p1.y)/(p2.x-p1.x);

        k=(p1.y+(m\*(x-p1.x)));

        temp.y=k;

        temp.x=x;

        for(i=0;i<4;i++)

            temp.code[i]=p1.code[i];

        if(temp.y<=350 && temp.y>=100)

            return (temp);

    }

    if(p1.code[0]=='1')

        y=100;

    if(p1.code[1]=='1')

        y=350;

    if((p1.code[0]=='1') || (p1.code[1]=='1'))

    {

        m=(float)(p2.y-p1.y)/(p2.x-p1.x);

        k=(float)p1.x+(float)(y-p1.y)/m;

        temp.x=k;

        temp.y=y;

        for(i=0;i<4;i++)

            temp.code[i]=p1.code[i];

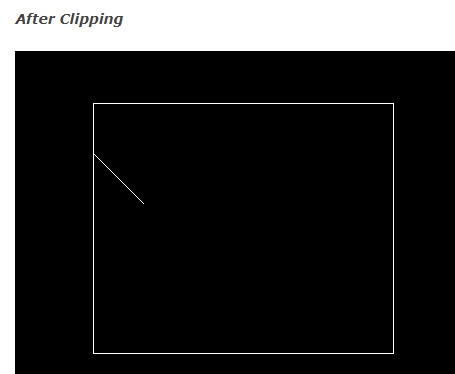
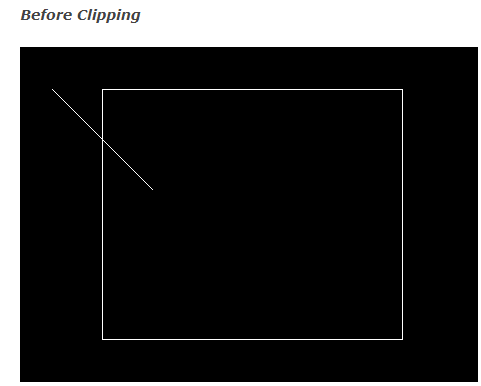
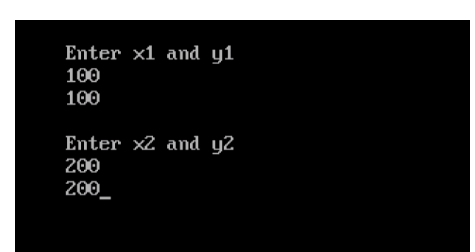
        return(temp);

    }

    else

        return(p1);

}



**Experiment No.11: 2D Geometric translation**

**Source Code:**

void translate(int,int);

void scale(int,int);

void rotate(float);

void main()

{

int ch;

int gd=DETECT,gm;

int tx,ty,sx,sy;

float theta;

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

setbkcolor(WHITE);

setcolor(6);

outtextxy (100,88,"Object.");

rectangle(100,250,150,200);

printf("---MENU---");

printf("\n 1)Translate\n 2)Scale\n 3)Rotate");

printf("\nEnter your choice: ");

scanf("%d",&ch);

cleardevice();

switch(ch)

{

case 1:

outtextxy(10,45,"Enter value of tx and ty:");

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

translate(tx,ty);

break;

case 2:

outtextxy(10,45,"Enter the value of sx and sy:");

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

scale(sx,sy);

break;

case 3:

outtextxy(10,50,"Enter the angle for rotation: ");

scanf("%f",&theta);

rotate(theta);

break;

default: printf("you have enterd wrong choice");

break;

}

getch();

closegraph();

}

void translate(int tx,int ty)

{

setcolor(2);

outtextxy(240,10,"TRANSLATION");

outtextxy(238,20,"------------");

rectangle(100,250,150,200);

rectangle(100+tx,250+ty,150+tx,200+ty);

}void scale(int sx,int sy)

{

setcolor(2);

outtextxy(240,10,"SCALING");

outtextxy(238,20,"--------");

rectangle(100,250,150,200);

rectangle(100\*sx,250\*sy,150\*sx,200\*sy);

}

void rotate(float theta)

{

int x1,x2,x3,x4;

int y1,y2,y3,y4;

int ax1,ax2,ax3,ax4,ay1,ay2,ay3,ay4;

int refx,refy;

theta=theta\*(3.14/180);

setcolor(2);

outtextxy(240,10,"ROTATE");

outtextxy(238,20,"-------");

refx=100;

refy=100;

x1=100;

y1=100;

x2=150;

y2=100;

x3=150;

y3=150;

x4=100;

y4=150;

ax1=refy+(x1-refx)\*cos(theta)-(y1-refy)\*sin(theta);

ay1=refy+(x1-refx)\*sin(theta)+(y1-refy)\*cos(theta);

ax2=refy+(x2-refx)\*cos(theta)-(y2-refy)\*sin(theta);

ay2=refy+(x2-refx)\*sin(theta)+(y2-refy)\*cos(theta);

ax3=refy+(x3-refx)\*cos(theta)-(y3-refy)\*sin(theta);

ay3=refy+(x3-refx)\*sin(theta)+(y3-refy)\*cos(theta);

ax4=refy+(x4-refx)\*cos(theta)-(y4-refy)\*sin(theta);

ay4=refy+(x4-refx)\*sin(theta)+(y4-refy)\*cos(theta);

rectangle(100,150,150,100);

line(ax1,ay1,ax2,ay2);

line(ax2,ay2,ax3,ay3);

line(ax3,ay3,ax4,ay4);

line(ax4,ay4,ax1,ay1);

}

**Output**

