INDEX

Computer Graphics

Aim : To study various functions used in Graphics Program implemented in C/C++

Header file **< graphics.h>**

C graphics using graphics.h functions can be used to draw different shapes, display text in different fonts, change colors and many more. Using functions of graphics.h in turbo c compiler we can make graphics programs, animations, projects and games. We can draw circles, lines, rectangles, bars and many other geometrical figures. You can change their colors using the available functions and fill them.

Graphics mode Initialization:

initgraph() function is used to load the graphics drivers and initialize the graphics system. For every function, that uses graphics mode, graphics mode must be initialized before using that function.

Declaration : -

```
void initgraph(int *graphdriver, int *graphmode, char *pathtodriver);
```

Sample Code :

```
#include<graphics.h>
#inciude<conio.h>

int main()
{
 int gd = DETECT, gm;
initgraph(&gd, &gm, "C:\\TC\\BGI");
getch();
closegraph();
return 0;
}
```

BASIC GRAPHICS FUNCTIONS

1) INITGRAPH

Initializes the graphics system.

Declaration

Void initgraph(int *graphdriver)

Remarks

To start the graphic system, you must first call initgraph.
Initgraph initializes the graphic system by loading a graphics driver from disk (or validating a registered driver) then putting the system into graphics mode.

Initgraph also resets all graphics settings (color, palette, current position, viewport, etc) to their defaults then resets graph.

2) GETPIXEL, PUTPIXEL

Getpixel gets the color of a specified pixel.
Putpixel places a pixel at a specified point.

Decleration

Unsigned getpixel(int x, int y)
Void putpixel(int x, int y, int color)

Remarks

Getpixel gets the color of the pixel located at (x,y);
Putpixel plots a point in the color defined at (x, y).

Return value

Getpixel returns the color of the given pixel.
Putpixel does not return.

3) CLOSE GRAPH

Shuts down the graphic system.

Decleration

Void closegraph(void);

Remarks

Close graph deallocates all memory allocated by the graphic system.
It then restores the screen to the mode it was in before you called initgraph.

Return value

None.

4) ARC, CIRCLE, PIESLICE

arc draws a circular arc.
Circle draws a circle
Pieslice draws and fills a circular pieslice

Decleration

Void arc(int x, int y, int stangle, int endangle, int radius);
Void circle(int x, int y, int radius);
Void pieslice(int x, int y, int stangle, int endangle, int radius);

Remarks

Arc draws a circular arc in the current drawing color
Circle draws a circle in the current drawing color
Pieslice draws a pieslice in the current drawing color, then fills it using the current fill pattern and fill color.

5) ELLIPSE, FILLELIPSE, SECTOR

Ellipse draws an elliptical arc.
Fillellipse draws and fills ellipse.

Decleration

Void ellipse(int x, int y, int stangle, int endangle, int xradius, int yradius)
Void fillellipse(int x, int y, int xradius, int yradius)

Remarks

Ellipse draws an elliptical arc in the current drawing color.
Fillellipse draws an elliptical arc in the current drawing color and than fills it with fill color and fill pattern.

6) FLOODFILL

Flood-fills a bounded region.

Declerat ion

Void floodfill(int x, int y, int border)

Remarks

Floodfills an enclosed area on bitmap device.
The area bounded by the color border is flooded with the current fill pattern and fill color.
(x,y) is a “seed point”
    If the seed is within an enclosed area, the inside will be filled.
    If the seed is outside the enclosed area, the exterior will be filled.
Use fillpoly instead of floodfill wherever possible so you can maintain code compatibility with future versions.
Floodfill doesnot work with the IBM-8514 driver.

Return value

If an error occurs while flooding a region, graph result returns ‘1’.

7) GETCOLOR, SETCOLOR

Getcolor returns the current drawing color.
Setcolor returns the current drawing color.

Decleration

Int getcolor(void);
Void setcolor(int color)

Remarks

Getcolor returns the current drawing color.
Setcolor sets the current drawing color to color, which can range from 0 to getmaxcolor.
To set a drawing color with setcolor , you can pass either the color number or the equivalent color name.

8) LINE, LINEREL,LINETO

Line draws a line between two specified pints.
Onerel draws a line relative distance from current position(CP).
Linrto draws a line from the current position (CP) to(x,y).

Decleration

Void lineto(int x, int y)

Remarks

Line draws a line from (x1, y1) to (x2, y2) using the current color, line style and thickness. It does not update the current position (CP).

Linerel draws a line from the CP to a point that is relative distance (dx, dy) from the CP, then advances the CP by (dx, dy).

Lineto draws a line from the CP to (x, y), then moves the CP to (x,y).

Return value

None

9) RECTANGLE

Draws a rectangle in graphics mode.

Decleration

Void far rectangle(int left, int top, int right, int bottom)

Remarks

It draws a rectangle in the current line style, thickness and drawing color.

(left, top) is the upper left corner of the rectangle, and (right, bottom) is its lower right corner.

Return value

None.

10) OUTTEXTXY()

outtextxy() outputs graphics text at the specified position (x,y) relative to the current viewport. The text is output using the current text font, text direction, character size and text justification settings. Its syntax is:

void far outtextxy(int x, int y, char far *tstring);

11) SETFILLSTYLE()

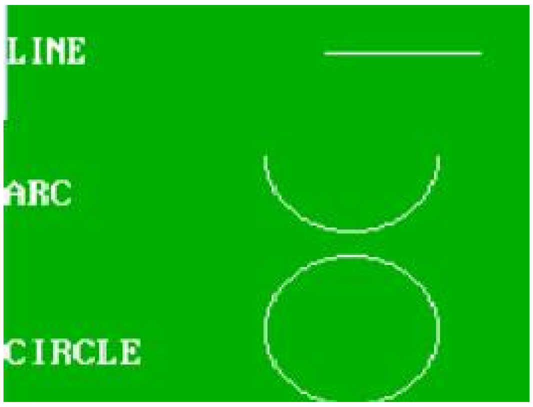
setfillstyle() sets the current fill pattern and fill color used by bar(), bar3D(), floodfill() and pieslice(). There are 11 predefined fill patterns. Its syntax is:

Program-1 Write a program to draw a circle line, arc, using inbuilt functions.

CODE:-

```
#include<stdio.h>
#include <graphics.h>
#include <conio.h>


int main()
{
    int gd = DETECT, gm;
     initgraph(&gd, &gm, "C:\\Turboc3\\BGI");  // Update the path to BGI folder as needed
setcolor(RED);
    circle(200, 200, 50);
   setcolor(GREEN);
   line(100, 100, 300, 300);
  setcolor(BLUE);
    arc(200, 200, 0, 180, 100);
    getch();
    closegraph();
    return 0;
}
```

Output:

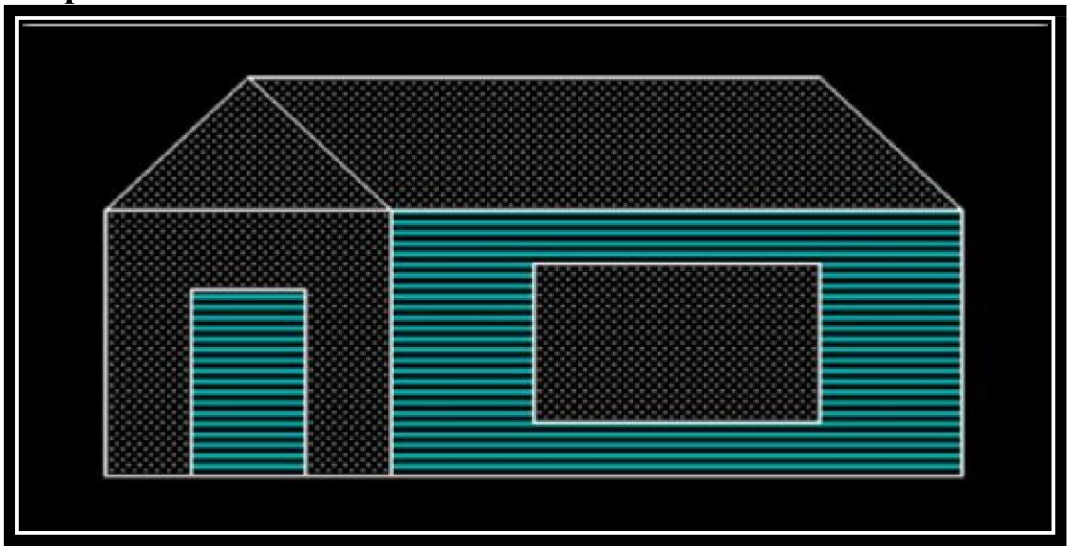


Program-2 Develop a home using graphics in built functions.

CODE:-

```
#include <conio.h>
#include <graphics.h>
#include <stdio.h>

void main()
{

int gdriver = DETECT, gmode;
initgraph(&gdriver, &gmode, "");

line(100, 100, 150, 50);
line(150, 50, 200, 100);
line(150, 50, 350, 50);
line(350, 50, 400, 100);
rectangle(100, 100, 200, 200);
rectangle(200, 100, 400, 200);
rectangle(130, 130, 170, 200);
rectangle(250, 120, 350, 180);

setfillstyle(2, 3);
floodfill(131, 131, WHITE);
floodfill(201, 101, WHITE);
setfillstyle(11, 7);
floodfill(101, 101, WHITE);
floodfill(150, 52, WHITE);
floodfill(163, 55, WHITE);
floodfill(251, 121, WHITE);

closegraph();
}
```

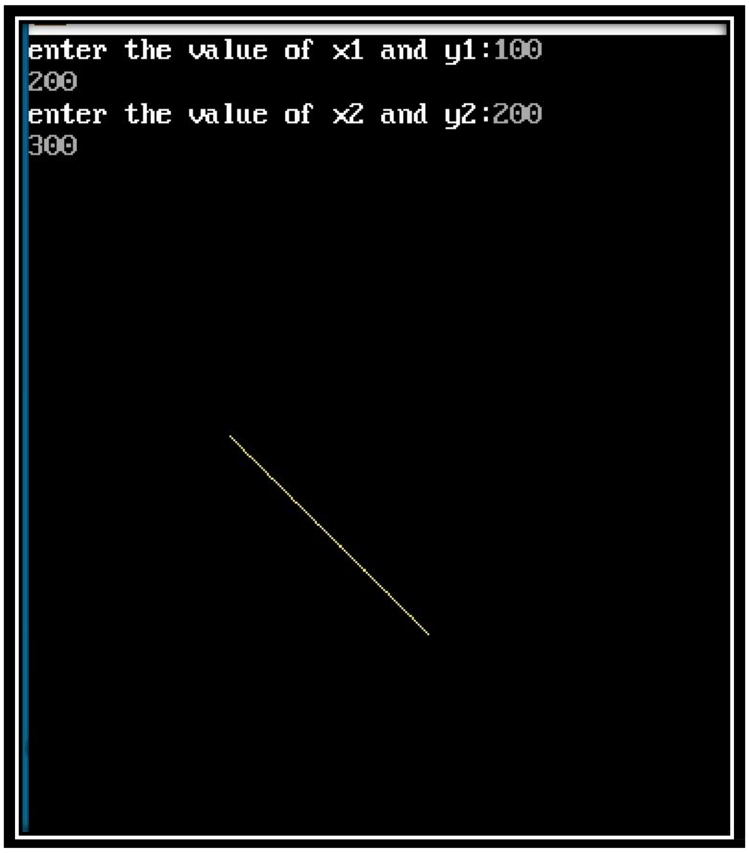
Output:-



Program-3 Write a program to draw a line using DDA line generating algorithm.
CODE:-

```
#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 & y1 :");
scanf("%f%f",&x1 , &y1);
printf("Enter the value of x2 & 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,YELLOW);
y=y+dy;
i=i+1;
delay(100);
}
getch();
closegraph();
}
```

Output:

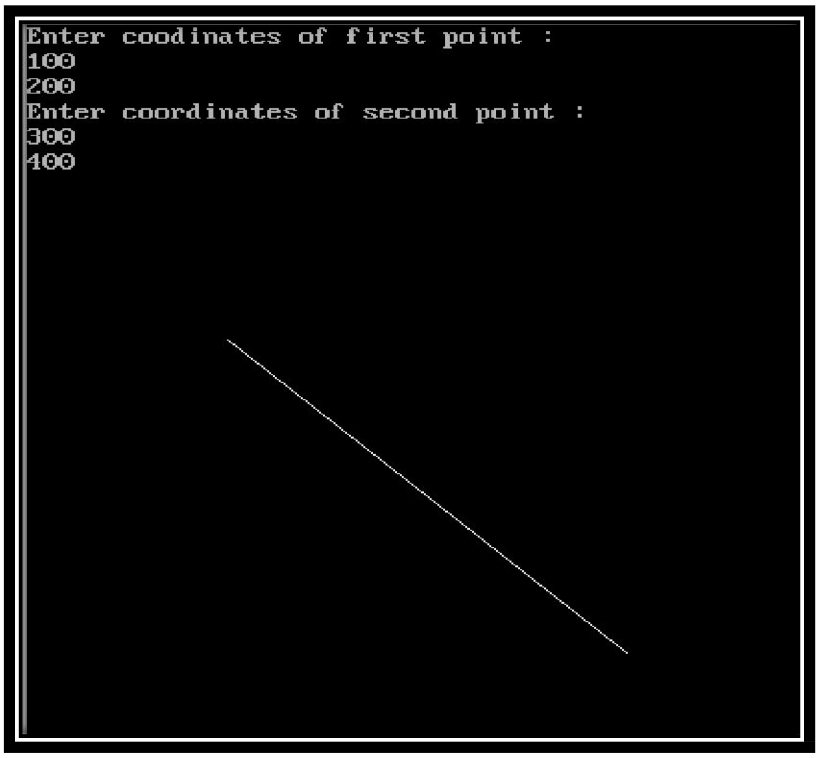


```
enter the value of x1 and y1:100
200
enter the value of x2 and y2:200
300
```

Program-4 Write a program to draw a line using Bradenham’s line generating algorithm.
CODE:-

```
#include<graphics.h>
#include<stdio.h>
#include<conio.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,WHITE);
y=y+1;
p=p+2*dy-2*dx;
}
else{ putpixel(x,y,WH
ITE); p=p+2*dy;
}
x=x+1;
}
}
int main(){
int gd=DETECT,gm,x0,y0,x1,y1;
initgraph(&gd,&gm,"C:\\TurboC3\\BGI");
printf("Enter coodinates of first point :");
scanf("%d%d",&x0 , &y0);
printf("Enter coordinates of second point :");
scanf("%d%d",&x1 , &y1);
drawline(x0,y0,x1,y1);
getch();
closegraph();
}
```

Output:-

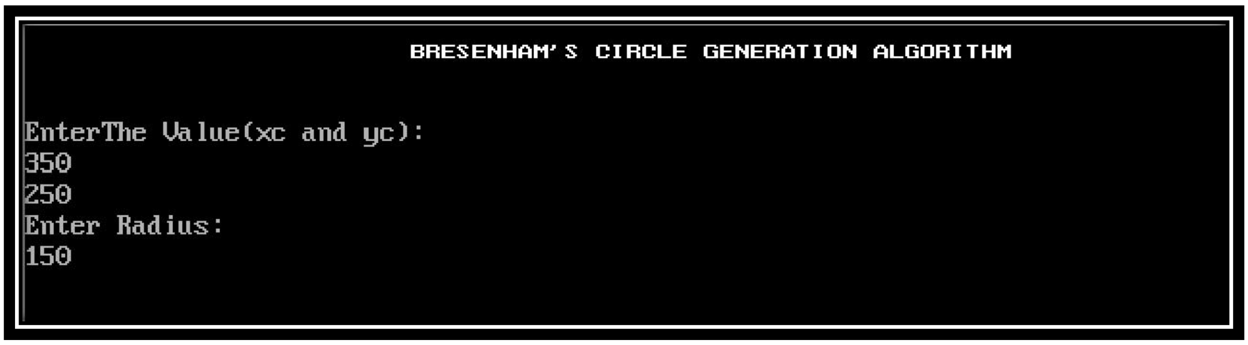


```
Enter coodinates of first point :
100
200
Enter coordinates of second point :
300
400
```

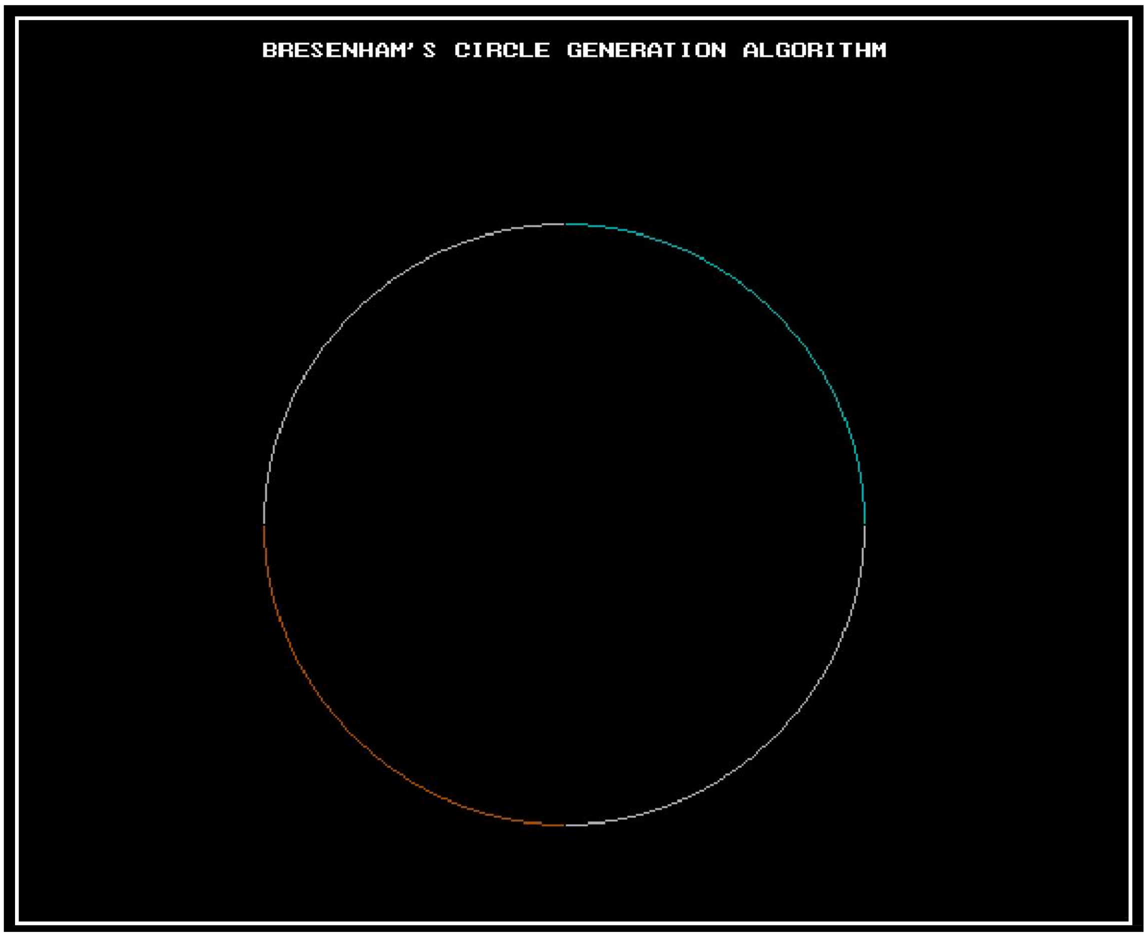
Program-5 Write a program to generate a circle using Bradenham’s circle drawing algorithm.
CODE:-

```
#include <stdio.h>
#include <conio.h>
#include <graphics.h>

void circlePlotPoint(int xc, int yc, int x, int y)
{
putpixel(xc + x, yc + y, 7);
putpixel(xc - x, yc + y, 6);
putpixel(xc + x, yc - y, 3);
putpixel(xc - x, yc - y, 7);
putpixel(xc + y, yc + x, 7);
putpixel(xc - y, yc + x, 6);
putpixel(xc + y, yc - x, 3);
putpixel(xc - y, yc - x, 7);
}
main()
{
int gd = DETECT, gm, x = 0, radius, xc, yc, y, p;
initgraph(&gd, &gm,"C:\\TurboC3\\BGI");
setbkcolor(BLACK);
// setcolor(BLUE);
outtextxy(200, 10, "BRESENHAM'S CIRCLE GENERATION ALGORITHM");
printf("\n\n");
printf("\nEnterThe Value(xc and yc): ");
scanf("%d %d", &xc, &yc);
printf("Enter Radius: ");
scanf("%d", &radius);
cleardevice();
outtextxy(200, 10, "BRESENHAM'S CIRCLE GENERATION ALGORITHM");
y = radius;
p = 3 - 2 * radius;
while (x < y)
{ x+
+;
if (p < 0)
{
p = p + 4 * x + 6;
}
else
{
y--;
p = p + 4 * (x - y) + 10;
}
circlePlotPoint(xc, yc, x, y);
delay(50);
}
getch();
closegraph();
}
```

Output:-



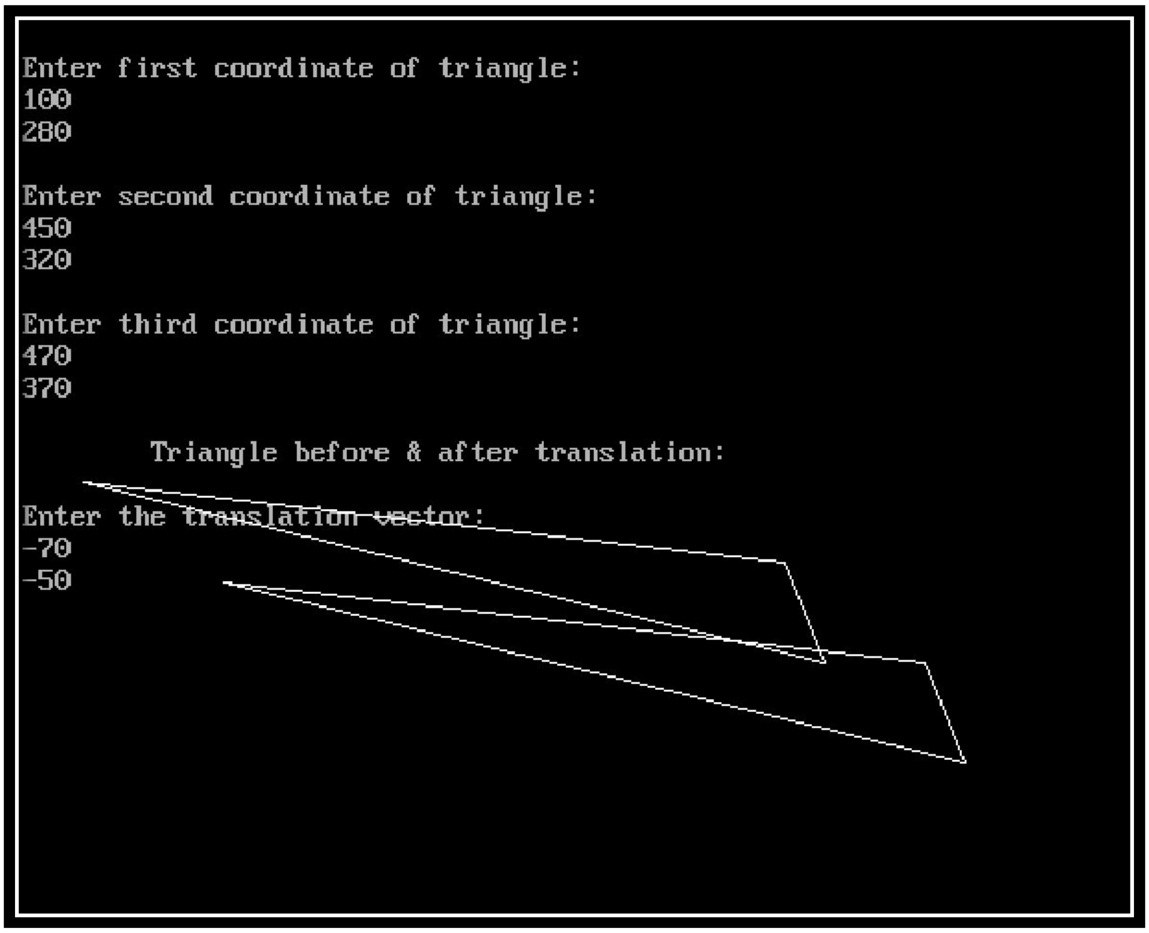
```
                    BRESENHAM'S CIRCLE GENERATION ALGORITHM

EnterThe Value(xc and yc):
350
250
Enter Radius:
150
```



Program-6 . Write a program to translate a triangle.
CODE:-

```
#include<stdio.h>
#include<conio.h>
#include<graphics.h>

void main()
{
int gd=DETECT,gm;
int x,y,x1,y1,x2,y2,tx,ty;

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

//setbkcolor(WHITE);

printf("\nEnter first coordinate of triangle: ");
scanf("%d%d",&x,&y);
printf("\nEnter second coordinate of triangle: ");
scanf("%d%d",&x1,&y1);
printf("\nEnter third coordinate of triangle: ");
scanf("%d%d",&x2,&y2);
printf("\n\tTriangle before & after translation: ");

line(x,y,x1,y1);
line(x1,y1,x2,y2);
line(x2,y2,x,y);
printf("\n\nEnter the translation vector: ");
scanf("%d%d",&tx,&ty);

// setcolor(RED);
line(x+tx,y+ty,x1+tx,y1+ty);
line(x1+tx,y1+ty,x2+tx,y2+ty);
ine(x2+tx,y2+ty,x+tx,y+ty);

getch();
closegraph();
}
```

Output:-

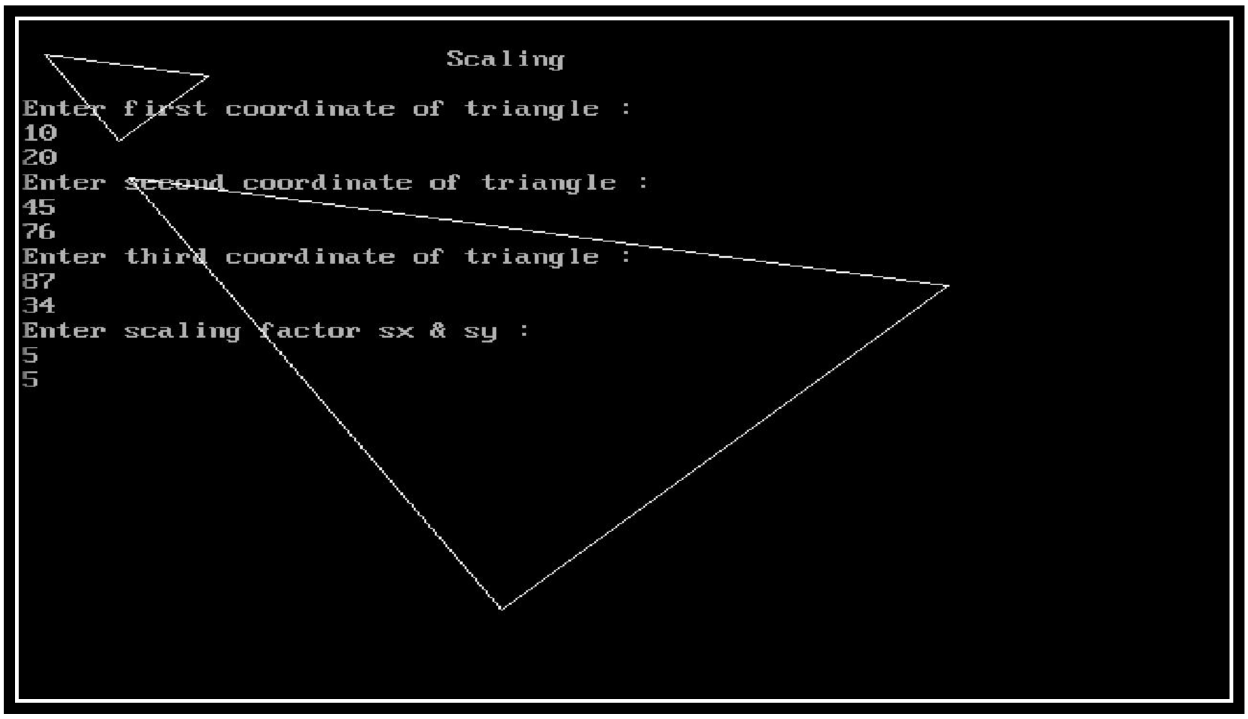


```
Enter first coordinate of triangle:
100
280

Enter second coordinate of triangle:
450
320

Enter third coordinate of triangle:
470
370

        Triangle before & after translation:

Enter the translation vector:
-70
-50
```

Program-7 Write a program to scale a triangle in 2-D.
CODE:-

```
#include<stdio.h>
#include<conio.h>
#include<graphics.h>
void main()
{
int x,y,x1,y1,x2,y2;
int sx,sy;
int gd=DETECT,gm;
initgraph(&gd,&gm,"C:\\TurboC3\\BGI");
printf("\n\t\t\t Scaling \n\n");
printf("Enter first coordinate of triangle :");
scanf("%d%d",&x,&y);
printf("Enter second coordinate of triangle :");
scanf("%d%d",&x1,&y1);
printf("Enter third coordinate of triangle :");
scanf("%d%d",&x2,&y2);

line(x,y,x1,y1);
line(x1,y1,x2,y2);
line(x2,y2,x,y);
printf("Enter scaling factor x & y :");
scanf("%d %d",&sx,&sy);
x=x*sx;
x1=x1*sx;
x2=x2*sx;
y=y*sy;
y1=y1*sy;
y2=y2*sy;
line(x,y,x1,y1);
line(x1,y1,x2,y2);
line(x2,y2,x,y);
getch();
closegraph();
}
```

Output:-



```
                                                Scaling

Enter first coordinate of triangle :
10
20
Enter second coordinate of triangle :
45
76
Enter third coordinate of triangle :
87
34
Enter scaling factor sx & sy :
5
5
```

Program-8 . Write a program to perform 2-D transformation operation, translation, rotation and scaling.

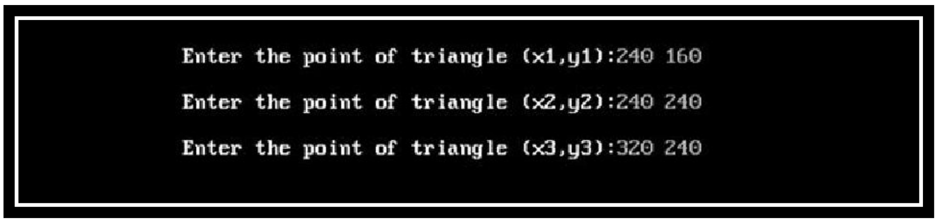
CODE:-

```
#include<conio.h>

#include<graphics.h>

#include<stdlib.h>

#include<stdio.h>

#include<math.h>


void translation();

void rotation();

void scalling();

void triangle();

void quadrant();

void main()

{

int gm;

int gd=DETECT;

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

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

printf("\nEnter the point of triangle (x1,y1):");

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

printf("\nEnter the point of triangle (x2,y2):");

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

printf("\nEnter the point of triangle (x3,y3):");

scanf("%d %d",&x3,&y3);

clrscr();

cleardevice();

setbkcolor(0);
```

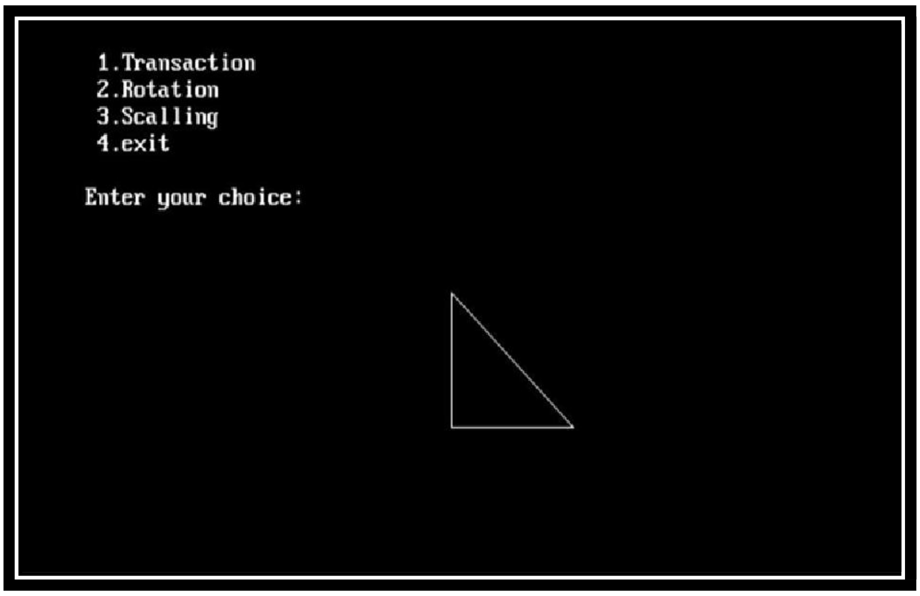
```
triangle(x1,y1,x2,y2,x3,y3);
while(1){
printf("\n 1.Transaction\n 2.Rotation\n 3.Scalling\n 4.exit \n");
printf("\nEnter your choice:");
scanf("%d",&c);
clrscr();
cleardevice();
quadrant();
switch(c){
   case 1:
   triangle(x1,y1,x2,y2,x3,y3);
   translation(x1,y1,x2,y2,x3,y3);
   getch();
   break;

   case 2:
   triangle(x1,y1,x2,y2,x3,y3);
   rotation(x1,y1,x2,y2,x3,y3);
   getch();
   break;

   case 3:
   triangle(x1,y1,x2,y2,x3,y3);
   scalling(x1,y1,x2,y2,x3,y3);
   getch();
   break;

   case 4:
   exit(1);
```

```
    break;
    default:
    printf("Enter the correct choice");
}
clrscr();
cleardevice();
getch()
}
```

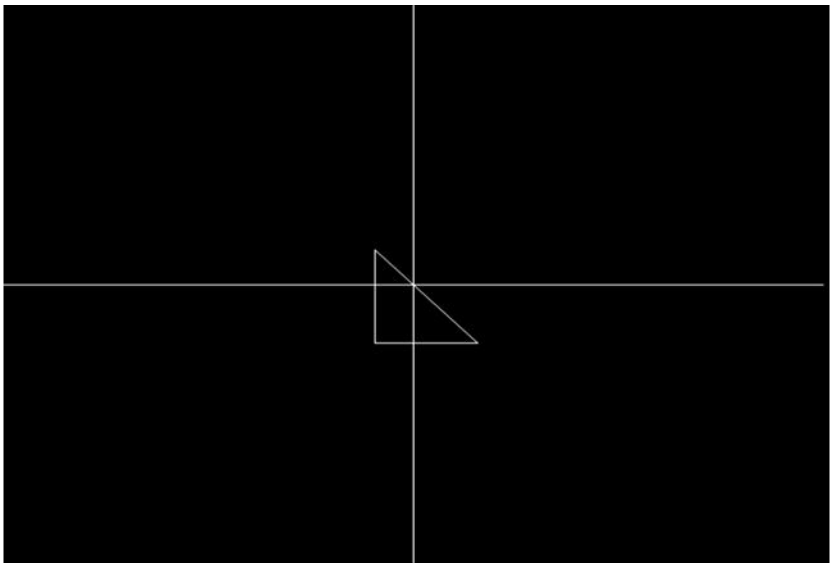
Output:-



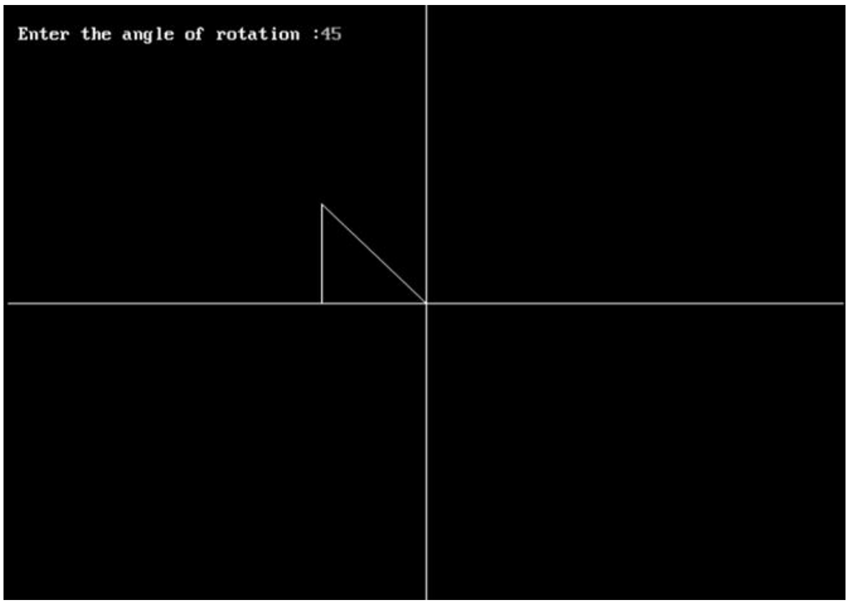
```
Enter the point of triangle (x1,y1):240 160
Enter the point of triangle (x2,y2):240 240
Enter the point of triangle (x3,y3):320 240
```



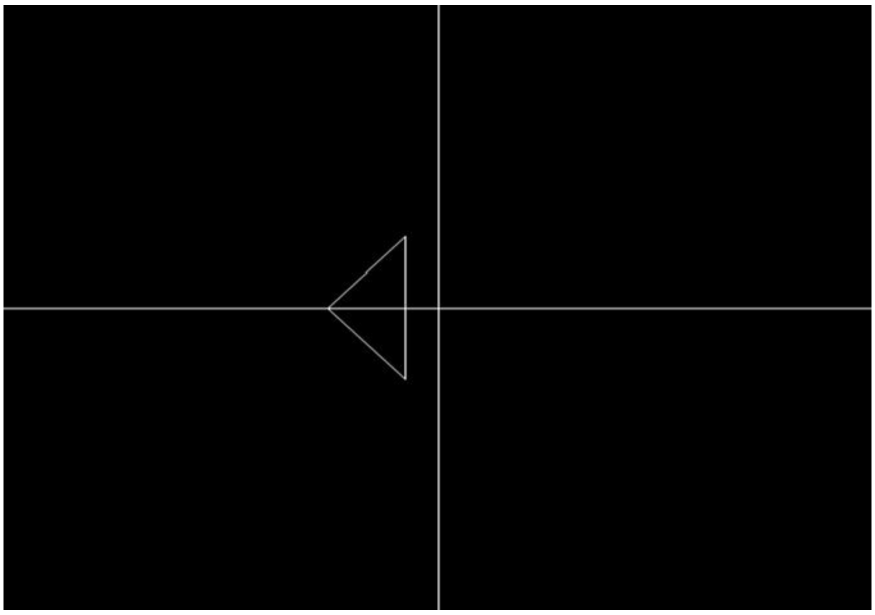
```
  1.Transaction
  2.Rotation
  3.Scalling
  4.exit

Enter your choice:
```

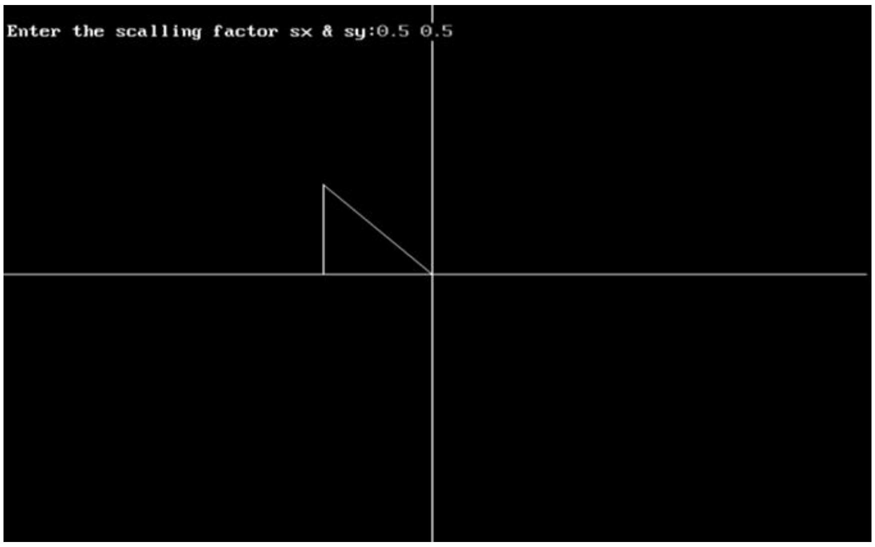


After Translation

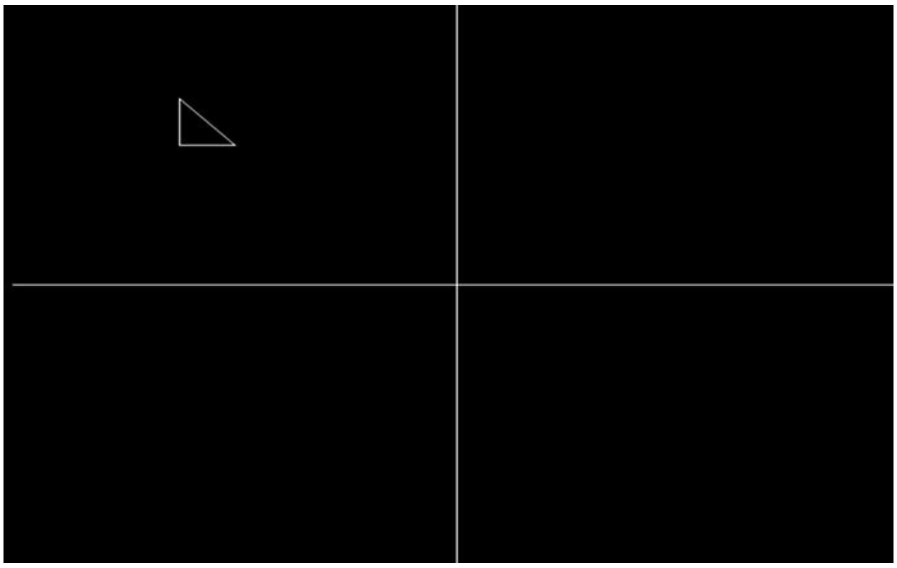


```
Enter the angle of rotation :45
```





```
Enter the scalling factor sx & sy:0.5 0.5
```



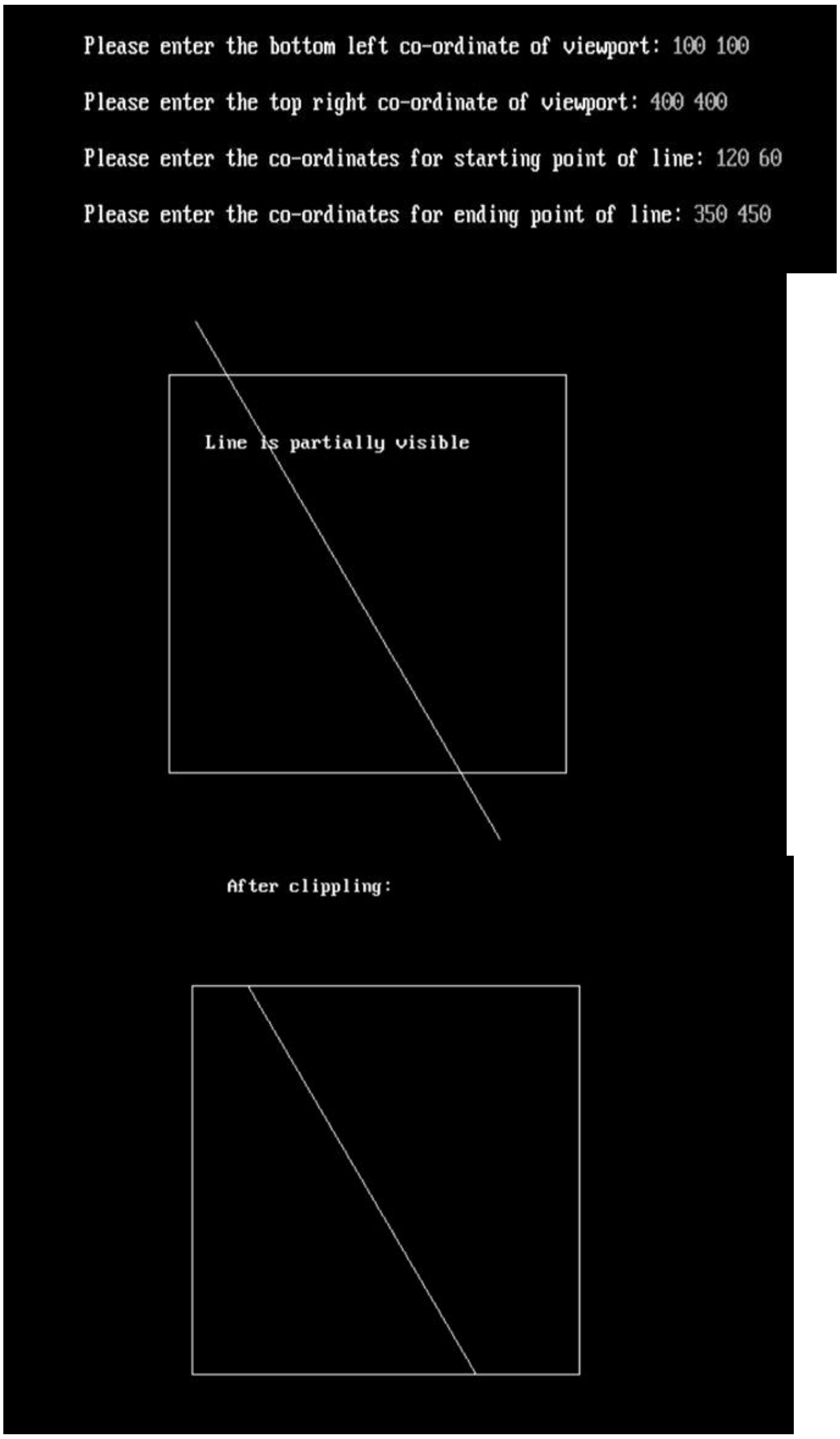
After Scaling

Program-9 Write a program for line clipping algorithm.
CODE:-

```
#include <stdio.h>
#include <conio.h>
#include <graphics.h>

void main() {
   int gd = DETECT, gm;
   float xmax, ymax, xmin, ymin, x1, y1, x2, y2, m;
   float start[4] = {0, 0, 0, 0}, end[4] = {0, 0, 0, 0}, code[4];
   clrscr();
   initgraph(&gd, &gm, "");  // Initialize graphics mode

   printf("\n\t Please enter the bottom left co-ordinate of viewport: ");
   scanf("%f %f", &xmin, &ymin);
   printf("\n\t Please enter the top right co-ordinate of viewport: ");
   scanf("%f %f", &xmax, &ymax);
   printf("\n\t Please enter the co-ordinates for starting point of line: ");
   scanf("%f %f", &x1, &y1);
   printf("\n\t Please enter the co-ordinates for ending point of line: ");
   scanf("%f %f", &x2, &y2);

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

   // Setting up the outcodes for the starting point
   if (x1 < xmin) start[0] = 1;
   if (x1 > xmax) start[1] = 1;
   if (y1 > ymax) start[2] = 1;
   if (y1 < ymin) start[3] = 1;

   // Setting up the outcodes for the ending point
   if (x2 < xmin) end[0] = 1;
   if (x2 > xmax) end[1] = 1;
   if (y2 > ymax) end[2] = 1;
   if (y2 < ymin) end[3] = 1;

   // Logical AND of the outcodes
   for (int i = 0; i < 4; i++) code[i] = start[i] && end[i];

   // Check for visibility
   if ((code[0] == 0) && (code[1] == 0) && (code[2] == 0) && (code[3] == 0)) {
      if ((start[0] == 0) && (start[1] == 0) && (start[2] == 0) && (start[3] == 0) &&
         (end[0] == 0) && (end[1] == 0) && (end[2] == 0) && (end[3] == 0))
         { cleardevice();
         printf("\n\t\tThe line is totally visible\n\t\tand not a clipping candidate");
         rectangle(xmin, ymin, xmax, ymax);
         line(x1, y1, x2, y2);
```

```
            getch();
        } else {
            cleardevice();
            printf("\n\t\tLine is partially visible");
            rectangle(xmin, ymin, xmax, ymax);
            line(x1, y1, x2, y2);
            getch();

            // Perform clipping
            if ((start[2] == 0) && (start[3] == 1)) { // Clipping bottom
                x1 = x1 + (ymin - y1) / m;
                y1 = ymin;
            }
            if ((end[2] == 0) && (end[3] == 1)) { // Clipping bottom
                x2 = x2 + (ymin - y2) / m;
                y2 = ymin;
            }
            if ((start[2] == 1) && (start[3] == 0)) { // Clipping top
                x1 = x1 + (ymax - y1) / m;
                y1 = ymax;
            }
            if ((end[2] == 1) && (end[3] == 0)) { // Clipping top
                x2 = x2 + (ymax - y2) / m;
                y2 = ymax;
            }
            if ((start[1] == 0) && (start[0] == 1)) { // Clipping left
                y1 = y1 + m * (xmin - x1);
                x1 = xmin;
            }
            if ((end[1] == 0) && (end[0] == 1)) { // Clipping left
                y2 = y2 + m * (xmin - x2);
                x2 = xmin;
            }
            if ((start[1] == 1) && (start[0] == 0)) { // Clipping right
                y1 = y1 + m * (xmax - x1);
                x1 = xmax;
            }
            if ((end[1] == 1) && (end[0] == 0)) { // Clipping right
                y2 = y2 + m * (xmax - x2);
                x2 = xmax;
            }

            cleardevice();
            printf("\n\t\tAfter clipping:");
            rectangle(xmin, ymin, xmax, ymax);
            line(x1, y1, x2, y2);
            getch();
        }
    } else {
        cleardevice();
        printf("\nLine is invisible");
        rectangle(xmin, ymin, xmax, ymax);
    }
    getch();
    closegraph();
}
```

Output:-

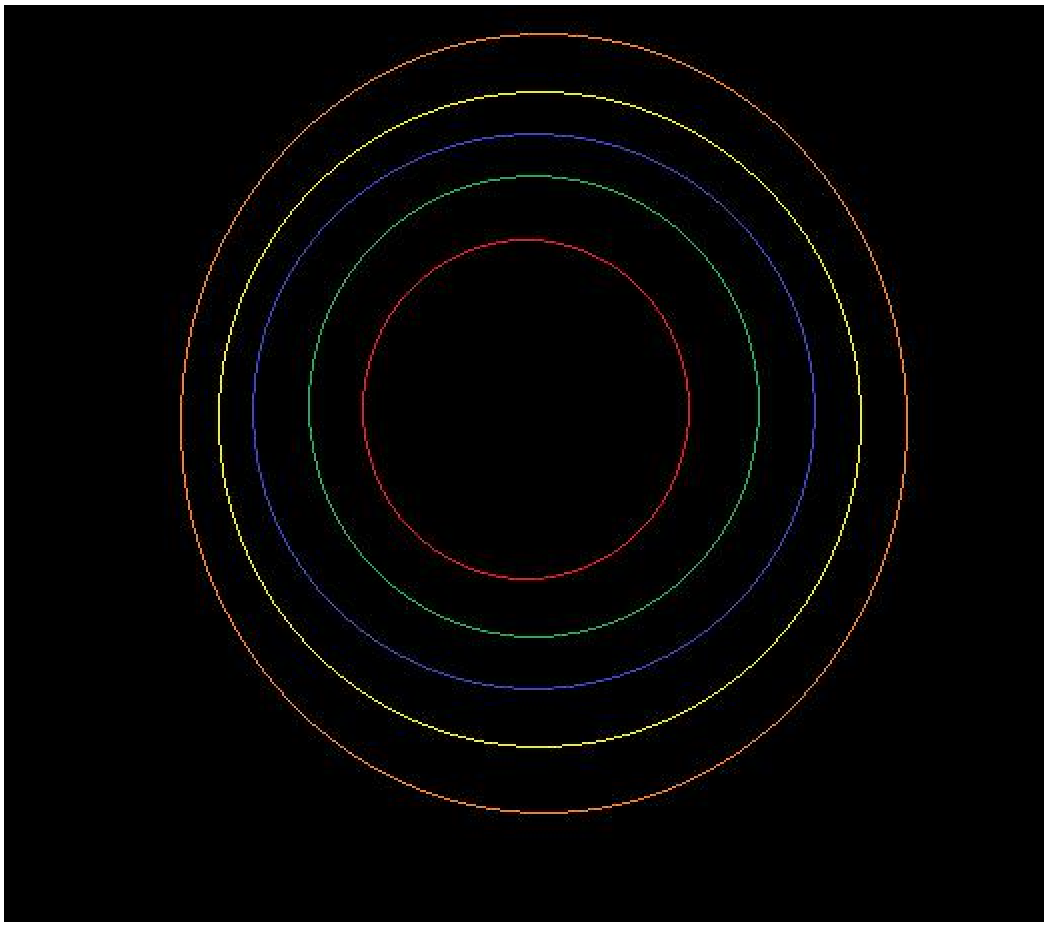


```
Please enter the bottom left co-ordinate of viewport: 100 100

Please enter the top right co-ordinate of viewport: 400 400

Please enter the co-ordinates for starting point of line: 120 60

Please enter the co-ordinates for ending point of line: 350 450

Line is partially visible

After clippling:
```

Program-10 Write a program to draw five circle with different color in c.
CODE:-

```
#include <graphics.h>
#include <conio.h>

int main() {
   int gd = DETECT, gm;
   int x = 100, y = 100, radius = 50;

   // Initialize graphics mode
   initgraph(&gd, &gm, "C:\\Turboc3\\BGI");

   // Set and draw five circles with different colors
   setcolor(RED);
   circle(x, y, radius);

   setcolor(GREEN);
   circle(x + 100, y, radius);

   setcolor(BLUE);
   circle(x + 200, y, radius);

   setcolor(YELLOW);
   circle(x + 300, y, radius);

   setcolor(ORANGE);
   circle(x + 400, y, radius);

   getch();
   closegraph();
   return 0;
}
```

Output:-

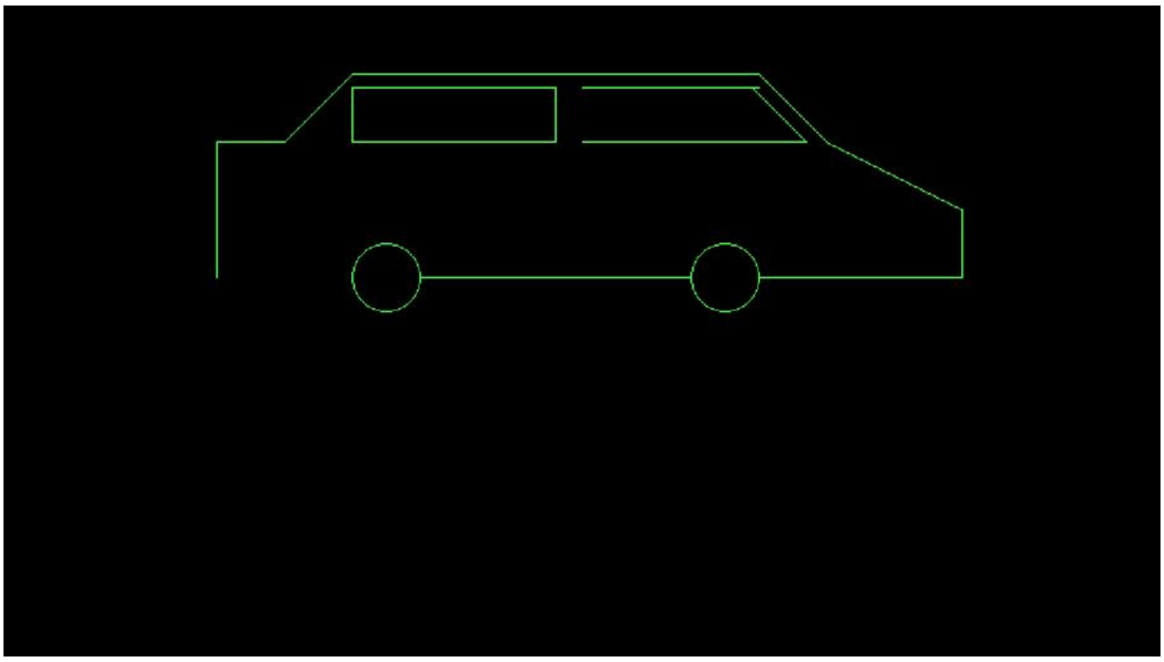


Program-11 Write a program to draw a car.

CODE:-

```
#include <conio.h>
#include <stdio.h>
#include <graphics.h>
#include <dos.h>

void main()
   { clrscr();
   int gd = DETECT, gm;
   initgraph(&gd, &gm, "c:\\TurboC3\\bgi");

   for (int i = 0; i <= 10; i++)
      { delay(300);
      setcolor(i);

      // Outer border of the shape
      line(100, 100, 150, 50);
      line(150, 50, 450, 50);
      line(450, 50, 500, 100);
      line(500, 100, 600, 150);
      line(600, 150, 600, 200);
      line(600, 200, 450, 200);
      line(450, 200, 200, 200);
      line(200, 200, 50, 200);
      line(50, 200, 50, 100);
      line(50, 100, 100, 100);  // Closing the outer shape

      // Inner structure (labelled parts)
      line(150, 60, 300, 60);  // a
      line(300, 60, 300, 100);  // b
      line(150, 100, 300, 100); // c
      line(150, 100, 150, 60);  // d

      line(320, 60, 450, 60);   // e
      line(445, 60, 485, 100);  // i
      line(320, 100, 485, 100); // j

      // Circles
      circle(425, 200, 25);     // f
      circle(175, 200, 25);     // g
   }

   getch();
   closegraph();
}
```

Output:-

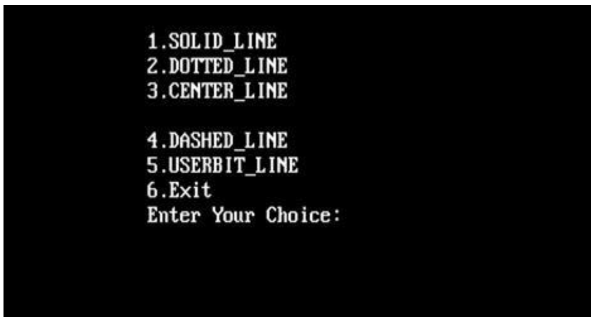


Program-12 Write a program to display different types of line.
CODE:-

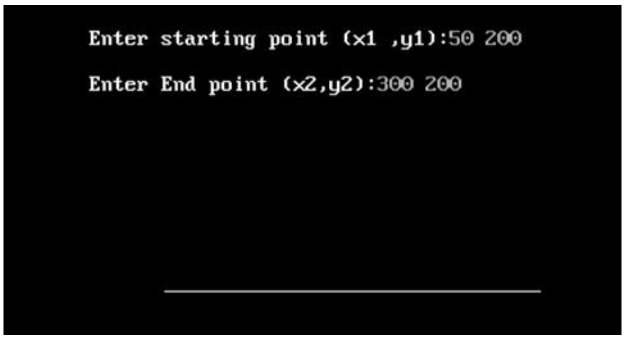
```
#include <stdio.h>
#include <conio.h>
#include <graphics.h>

void main() {
   int gd = DETECT, gm, c;
   int x1, x2, y1, y2;
   initgraph(&gd, &gm, "");

   setbkcolor(0);

   while (1) {
      printf("\n1. SOLID_LINE\n2. DOTTED_LINE\n3. CENTER_LINE\n");
      printf("4. DASHED_LINE\n5. USERBIT_LINE\n6. Exit");
      printf("\nEnter Your Choice: ");
      scanf("%d", &c);
      clrscr();
      cleardevice();

      if (c < 6) {
         printf("\nEnter starting point (x1, y1): ");
         scanf("%d %d", &x1, &y1);
         printf("\nEnter End point (x2, y2): ");
         scanf("%d %d", &x2, &y2);
      }

       switch (c)
        { case 1:
            setlinestyle(SOLID_LINE, 1, 1);
            setcolor(15);
            line(x1, y1, x2, y2);
            getch();
            cleardevice();
            break;

         case 2:
            setlinestyle(DOTTED_LINE, 1, 1);
            setcolor(15);
            line(x1, y1, x2, y2);
            getch();
            cleardevice();
            break;

         case 3:
            setlinestyle(CENTER_LINE, 1, 1);
            setcolor(15);
            line(x1, y1, x2, y2);
```

```
                getch();
                cleardevice();
                break;

            case 4:
                setlinestyle(DASHED_LINE, 1, 1);
                setcolor(15);
                line(x1, y1, x2, y2);
                getch();
                cleardevice();
                break;

            case 5:
                setlinestyle(USERBIT_LINE, 1, 1);
                setcolor(15);
                line(x1, y1, x2, y2);
                getch();
                cleardevice();
                break;

            case 6:
                closegraph();
                exit(0);

            default:
                printf("!Enter the correct choice!\n");
                break;
        }
        clrscr();
        cleardevice();
    }
}
```

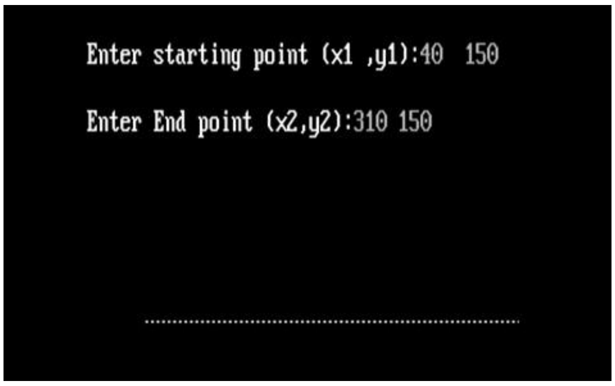
Output:-



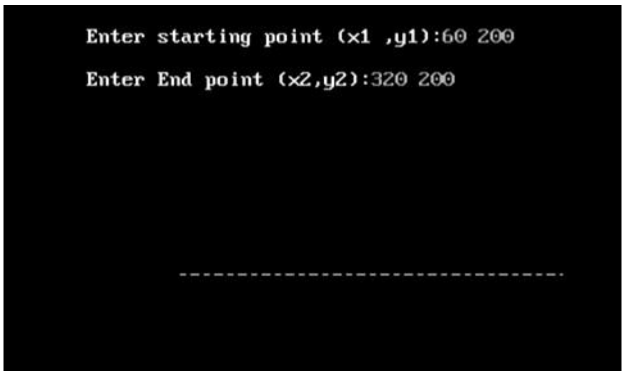
```
1.SOLID_LINE
2.DOTTED_LINE
3.CENTER_LINE

4.DASHED_LINE
5.USERBIT_LINE
6.Exit
Enter Your Choice:
```



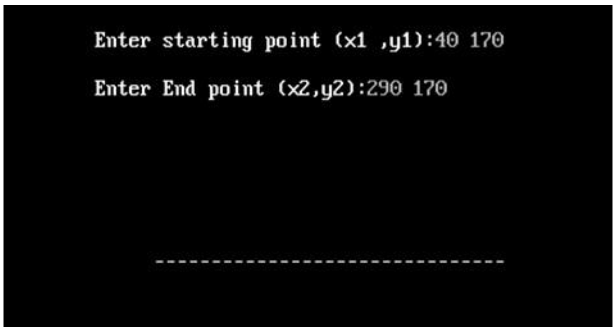
```
Enter starting point (x1 ,y1):50 200
Enter End point (x2,y2):300 200
```



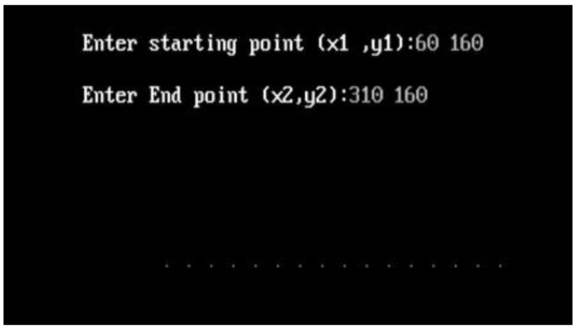
```
Enter starting point (x1 ,y1):40  150
Enter End point (x2,y2):310 150
```



```
Enter starting point (x1 ,y1):60 200
Enter End point (x2,y2):320 200
```



```
Enter starting point (x1 ,y1):40 170
Enter End point (x2,y2):290 170
```



```
Enter starting point (x1 ,y1):60 160
Enter End point (x2,y2):310 160
```

Program-13 Write a program to make a 3-D bar.

CODE:-

```
#include <stdio.h>

#include <conio.h>

#include <graphics.h>

int main() {

int gd = DETECT, gm;

clrscr();

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

 // Setting color and fill for the first bar

    setcolor(GREEN);

    setfillstyle(SOLID_FILL, RED);

    bar(20, 30, 200, 400);

// Setting color and fill for the 3D bar

    setfillstyle(LTSLASH_FILL, GREEN);

    bar3d(150, 100, 200, 200, 20, 1);

getch();

 closegraph();

 return 0;

}
```

Output:-