NAME: SINGH SUDHAM DHARMENDRA

ROLL NO.: AIMLD50

BRANCH: CSE (AI & ML)

SUBJECT: COMPUTER GRAPHICS

TOPIC: EXPERIMENT NO. 1

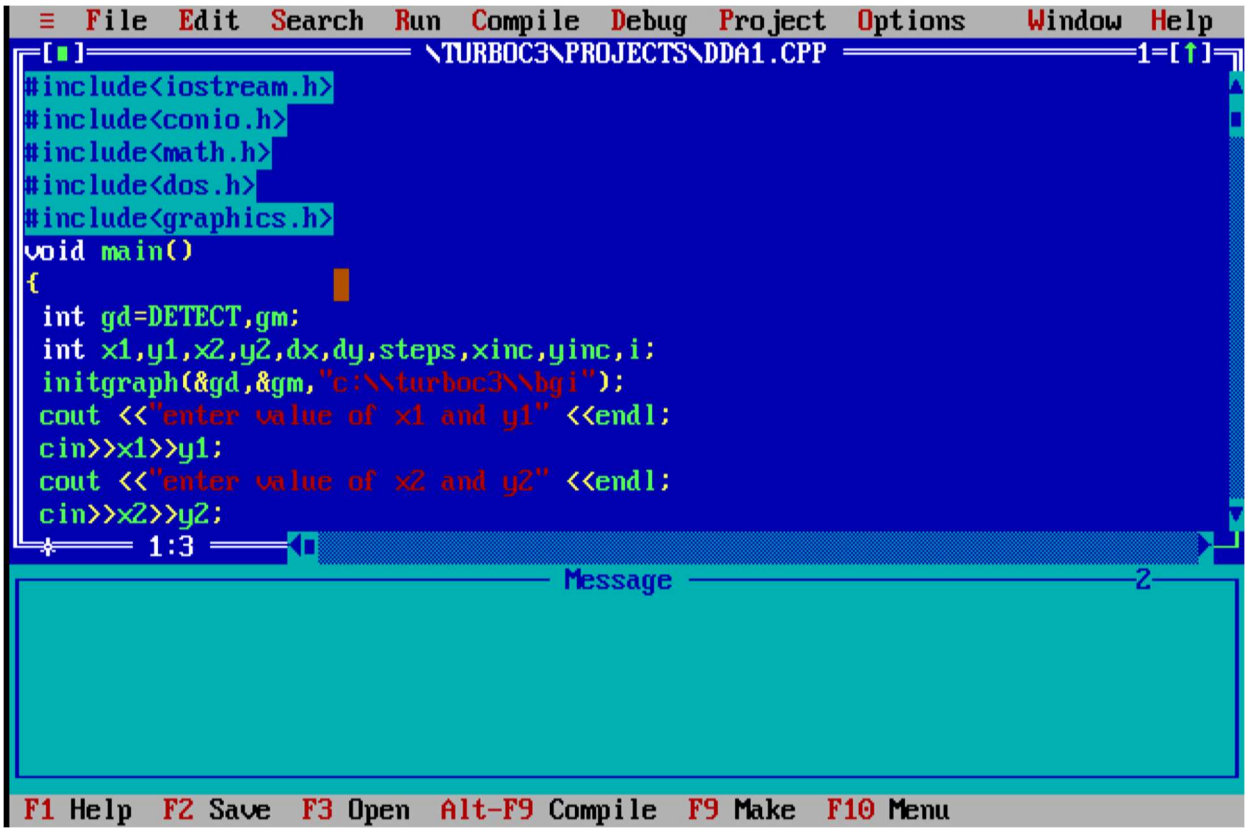
EXPERIMENT NO. 1

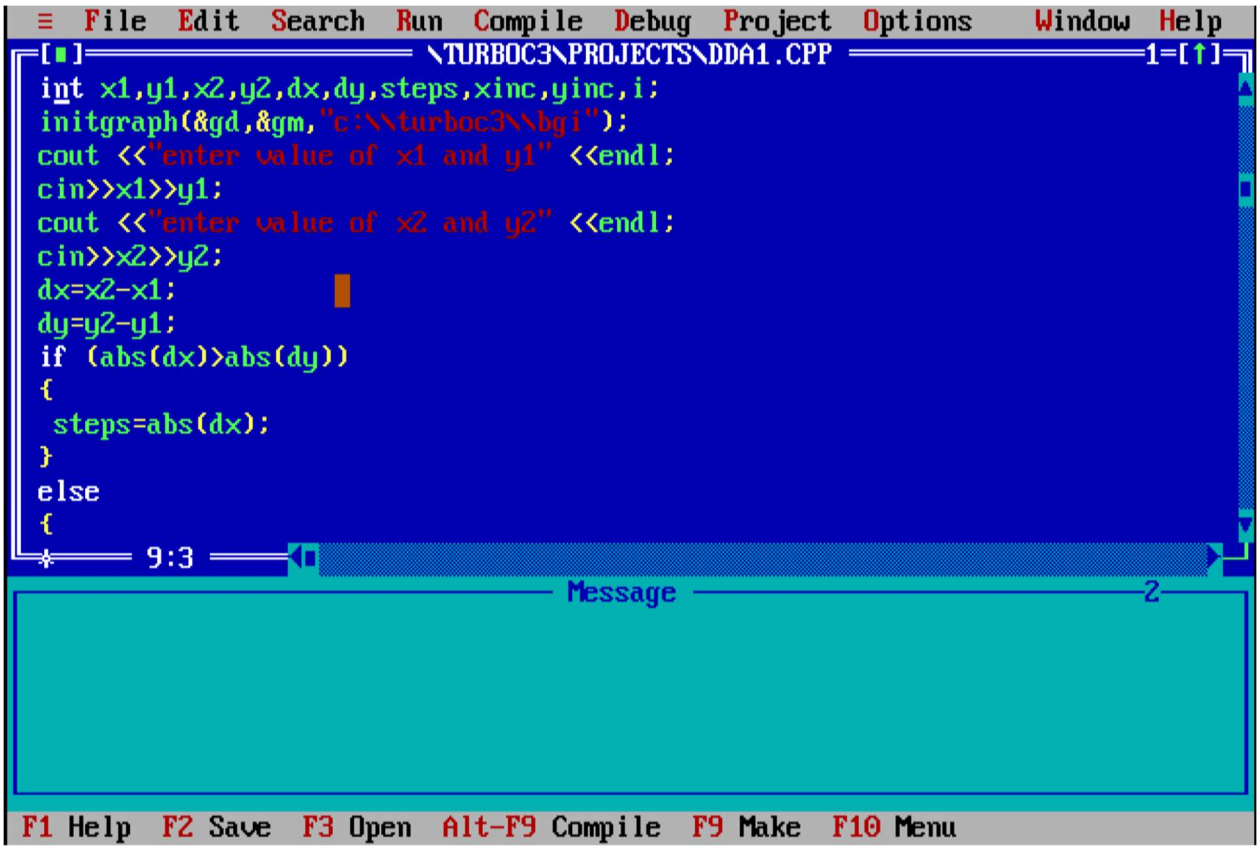
AIM: WRITE A PROGRAM IN C++ FOR DDA LINE DRAWING ALGORITHM.

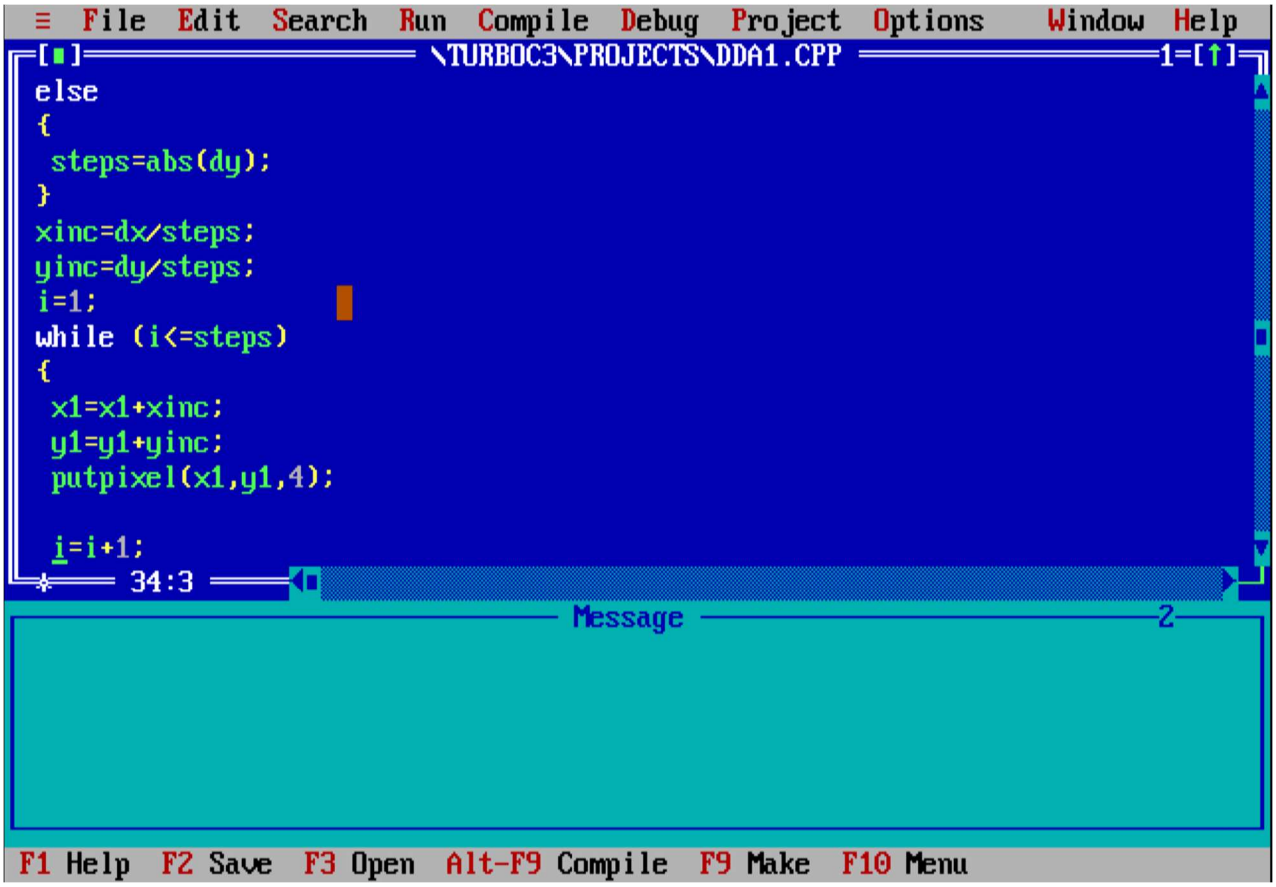
(i)TO DRAW A THIN LINE.

SOFTWARE USED: TURBO C++

SOURCE CODE:

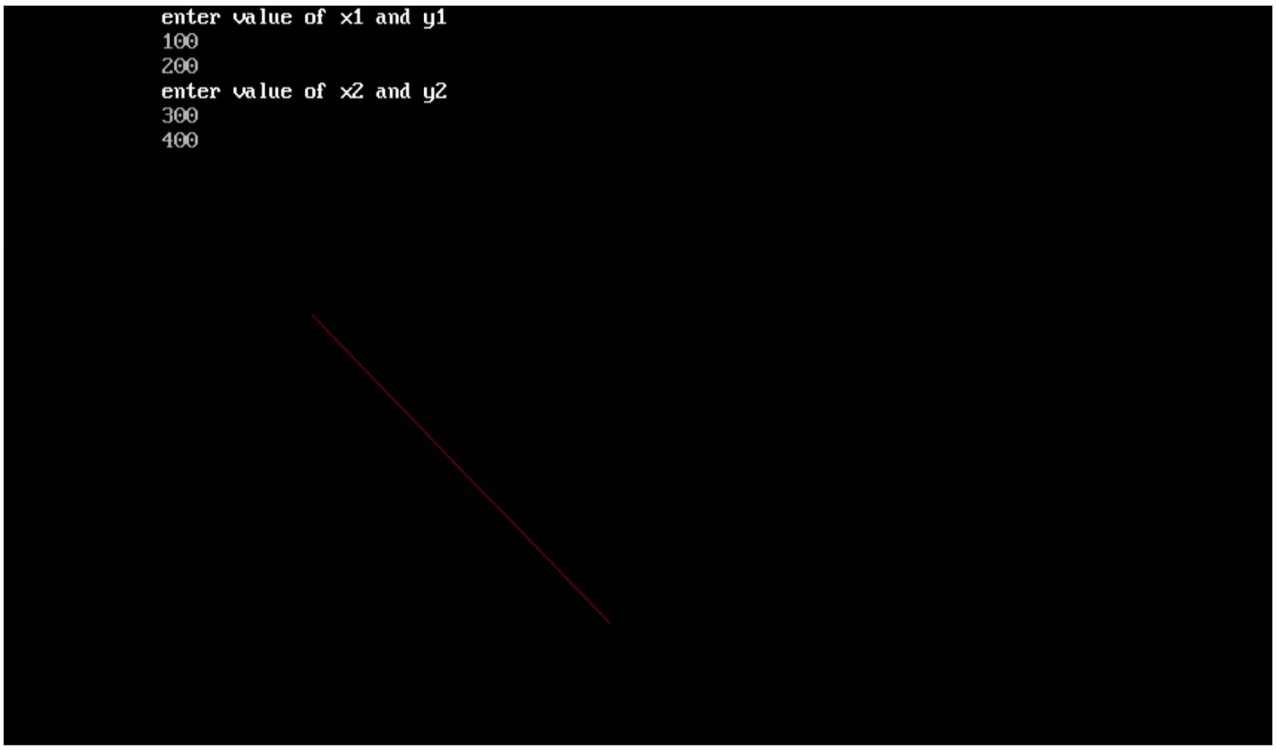

File Edit Search Run Compile Debug Project Options Window Help
\TURBOC3\PROJECTS\DDA1.CPP
1=[↑]
#include<iostream.h>
#include<conio.h>
#include<math.h>
#include<dos.h>
#include<graphics.h>
void main()
{
int gd=DETECT,gm;
int x1,y1,x2,y2,dx,dy,steps,xinc,yinc,i;
initgraph(&gd,&gm,"c:\\turboc3\\bgi");
cout <<"enter value of x1 and y1" <<endl;
cin>>x1>>y1;
cout <<"enter value of x2 and y2" <<endl;
cin>>x2>>y2;
1:3
Message
2
F1 Help F2 Save F3 Open Alt-F9 Compile F9 Make F10 Menu


File Edit Search Run Compile Debug Project Options Window Help
\TURBOC3\PROJECTS\DDA1.CPP
int x1,y1,x2,y2,dx,dy,steps,xinc,yinc,i;
initgraph(&gd,&gm,"c:\\turboc3\\bgi");
cout <<"enter value of x1 and y1" <<endl;
cin>>x1>>y1;
cout <<"enter value of x2 and y2" <<endl;
cin>>x2>>y2;
dx=x2-x1;
dy=y2-y1;
if (abs(dx)>abs(dy))
{
steps=abs(dx);
}
else
{
9:3
Message
F1 Help F2 Save F3 Open Alt-F9 Compile F9 Make F10 Menu


File Edit Search Run Compile Debug Project Options Window Help
\TURBOC3\PROJECTS\DDA1.CPP
1=[↑]
else
{
steps=abs(dy);
}
xinc=dx/steps;
yinc=dy/steps;
i=1;
while (i<=steps)
{
x1=x1+xinc;
y1=y1+yinc;
putpixel(x1,y1,4);
i=i+1;
34:3
Message
2
F1 Help F2 Save F3 Open Alt-F9 Compile F9 Make F10 Menu


File Edit Search Run Compile Debug Project Options Window Help
\TURBOC3\PROJECTS\DDA1.CPP
1=[↑]
i=1;
while (i<=steps)
{
x1=x1+xinc;
y1=y1+yinc;
putpixel(x1,y1,4);
i=i+1;
delay(50);
}
getch();
closegraph();
}
40:3
Message
2
F1 Help F2 Save F3 Open Alt-F9 Compile F9 Make F10 Menu

OUTPUT:



CONCLUSION:

WITH THE HELP OF TURBO C++ FOR DDA LINE DRAWING ALGORITHM, WE CAN TAKE INPUT FROM THE USER FOR X AND Y CO-ORDINATE AND CAN DRAW AN DESIRED STRAIGHT LINE.

NAME:- SINGH SUDHAM DHARMENDRA

BRANCH:- CSE(AIML)

ROLL NO.:- AIMLD50

SUBJECT:- COMPUTER GRAPHICS

TOPIC:-EXPERIMENT NO:-2

EXPERIMENT NO. 2

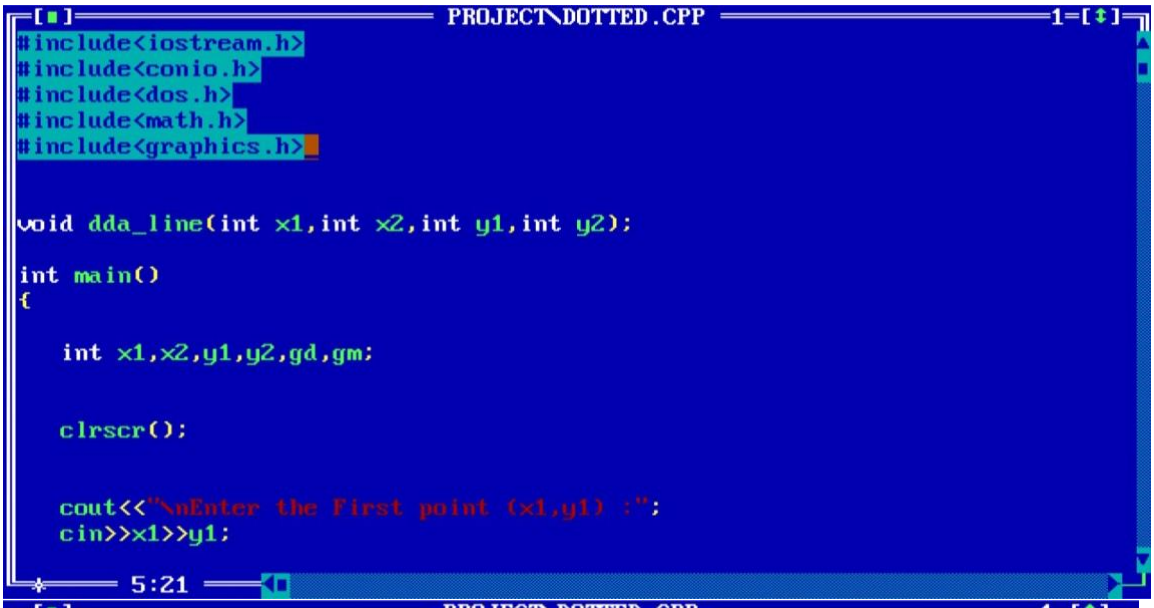
AIM: - A program in C/C++ for DDA line drawing algorithm.

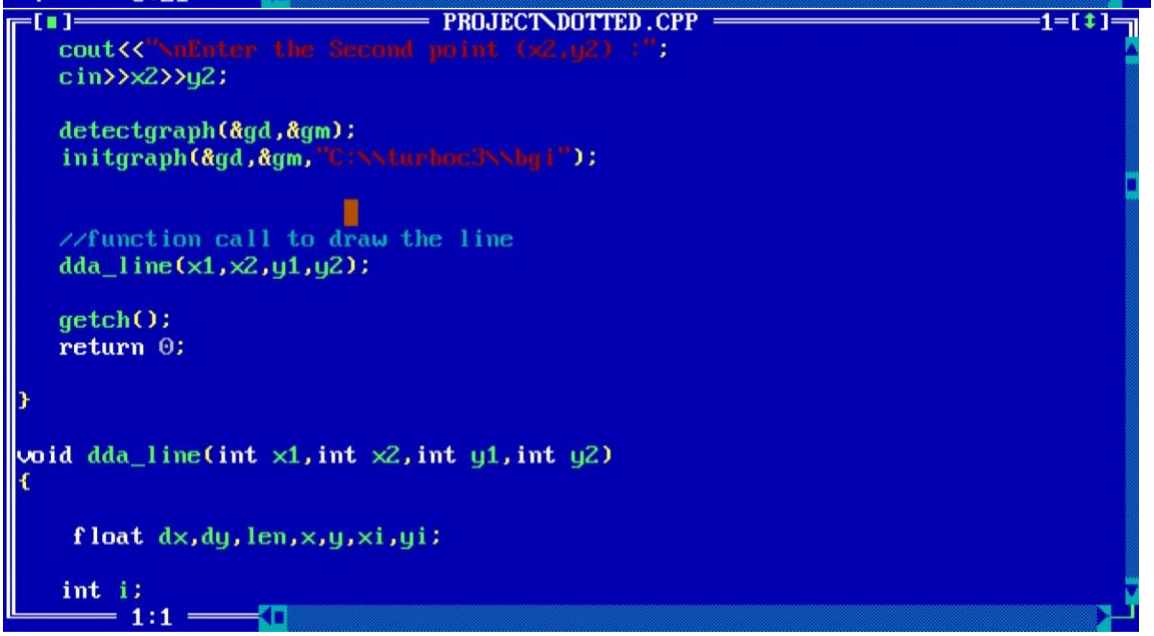
i) **To draw a dotted line.**

ii) **To draw a dashed line.**

SOFTWARE USED: - Turbo C++.

SOURCE CODE: - 1) To draw a dotted line.


PROJECT\DOTTED.CPP
1
#include<iostream.h>
#include<conio.h>
#include<dos.h>
#include<math.h>
#include<graphics.h>
void dda_line(int x1,int x2,int y1,int y2);
int main()
{
int x1,x2,y1,y2,gd,gm;
clrscr();
cout<<"\nEnter the First point (x1,y1) :";
cin>>x1>>y1;
5:21

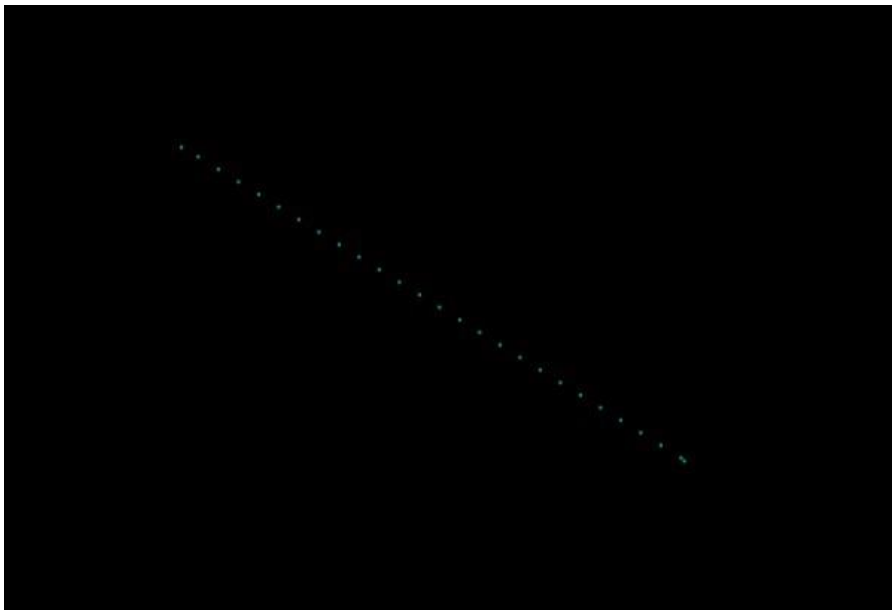

PROJECT\DOTTED.CPP
1
cout<<"\nEnter the Second point (x2,y2) :";
cin>>x2>>y2;
detectgraph(&gd,&gm);
initgraph(&gd,&gm,"C:\\turboc3\\bgi");
//function call to draw the line
dda_line(x1,x2,y1,y2);
getch();
return 0;
}
void dda_line(int x1,int x2,int y1,int y2)
{
float dx,dy,len,x,y,xi,yi;
int i;
1:1

```
PROJECT\DOTTED.CPP                                    1=[↕]
   dx = abs(x2-x1);

   dy = abs(y2-y1);


   if(dx>=dy)

      len=dx;

          else

          len=dy;


   dx = (x2-x1)/len;

   dy = (y2-y1)/len;

1:1
```

```
PROJECT\DOTTED.CPP                                    1=[↕]
   x = x1 + 0.5;

   y = y1 + 0.5;

   putpixel(x1,y1,3);

   putpixel(x2,y2,3);

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

    {  if(i%6>4)

       putpixel(x,y,3);

       x += dx;

       y += dy;

    }
}
1:1
```

OUTPUT: -

```
Enter the First point (x1,y1) :100
200

Enter the Second point (x2,y2) :250
300_
```

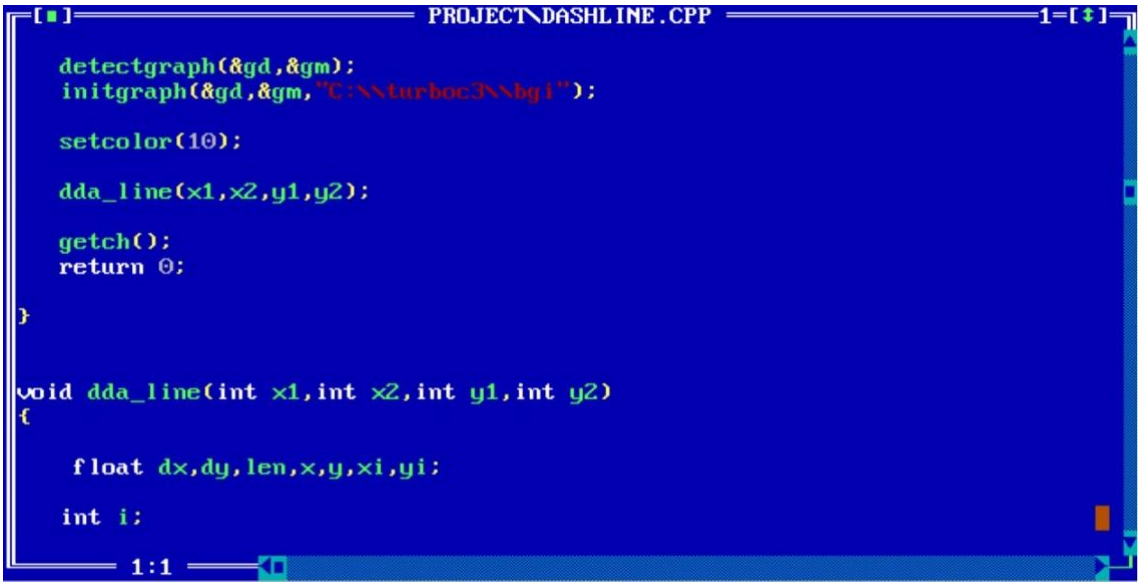


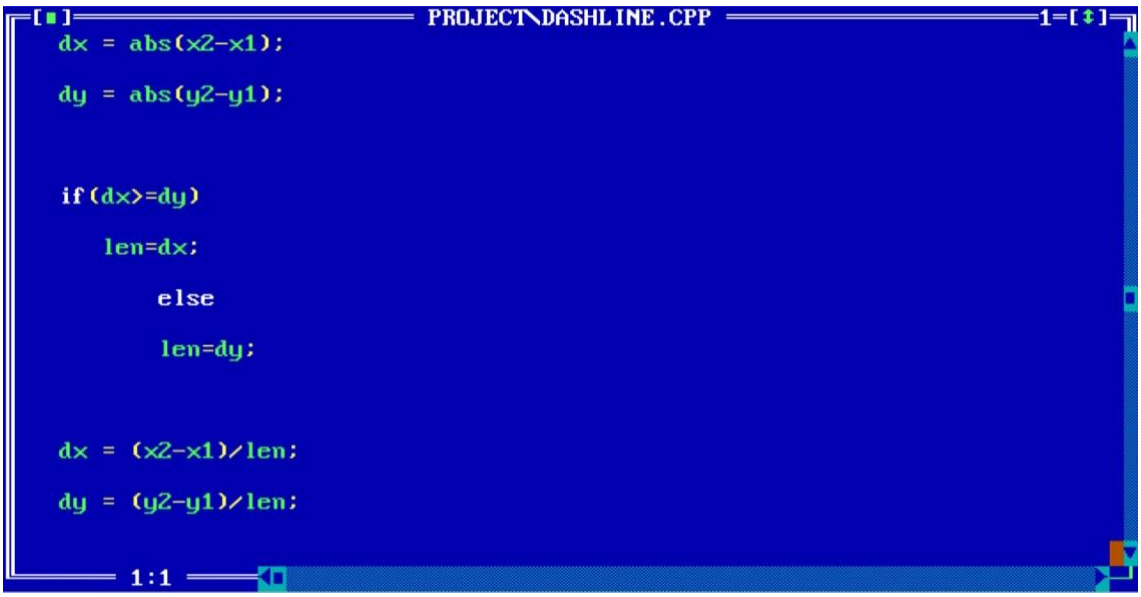
2) To draw dashed line.

PROJECT\DASHLINE.CPP

```
#include<iostream.h>
#include<conio.h>
#include<dos.h>
#include<math.h>
#include<graphics.h>


void dda_line(int x1,int x2,int y1,int y2);

int main()
{

   int x1,x2,y1,y2,gd,gm;

   clrscr();

   cout<<"\nEnter the First point (x1,y1) :";
   cin>>x1>>y1;

   cout<<"\nEnter the Second point (x2,y2) :";
   cin>>x2>>y2;
```

5:21

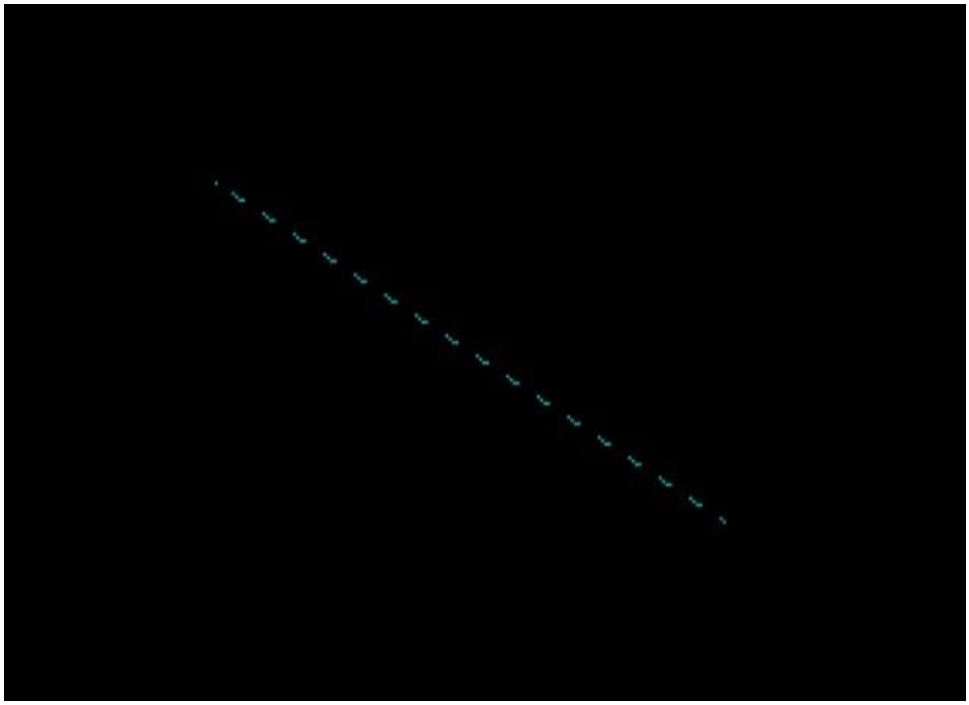

PROJECT\DASHLINE.CPP
1=[↕]
detectgraph(&gd,&gm);
initgraph(&gd,&gm,"C:\\turboc3\\bgi");
setcolor(10);
dda_line(x1,x2,y1,y2);
getch();
return 0;
}
void dda_line(int x1,int x2,int y1,int y2)
{
float dx,dy,len,x,y,xi,yi;
int i;
1:1


PROJECT\DASHLINE.CPP
1=[↕]
dx = abs(x2-x1);
dy = abs(y2-y1);
if(dx>=dy)
len=dx;
else
len=dy;
dx = (x2-x1)/len;
dy = (y2-y1)/len;
1:1

```
PROJECT\DASHLINE.CPP                                    1=[↕]
   x = x1 + 0.5;
   y = y1 + 0.5;
   putpixel(x1,y1,3);
   putpixel(x2,y2,3);
   for(i=0;i<=len;i++)
    {  if(i%9>4)
       putpixel(x,y,3);
       x += dx;
       y += dy;
    }
}
1:1
```

OUTPUT:-

```
Enter the First point (x1,y1) :100
200

Enter the Second point (x2,y2) :250
300_
```



CONCLUSION: -

Hence by taking input from user dotted and dashed line can be drawn by using DDA line drawing algorithm.

NAME:- SINGH SUDHAM DHARMENDRA

BRANCH:- CSE(AIML)

ROLL NO.:- AIMLD50

SUBJECT:- COMPUTER GRAPHICS

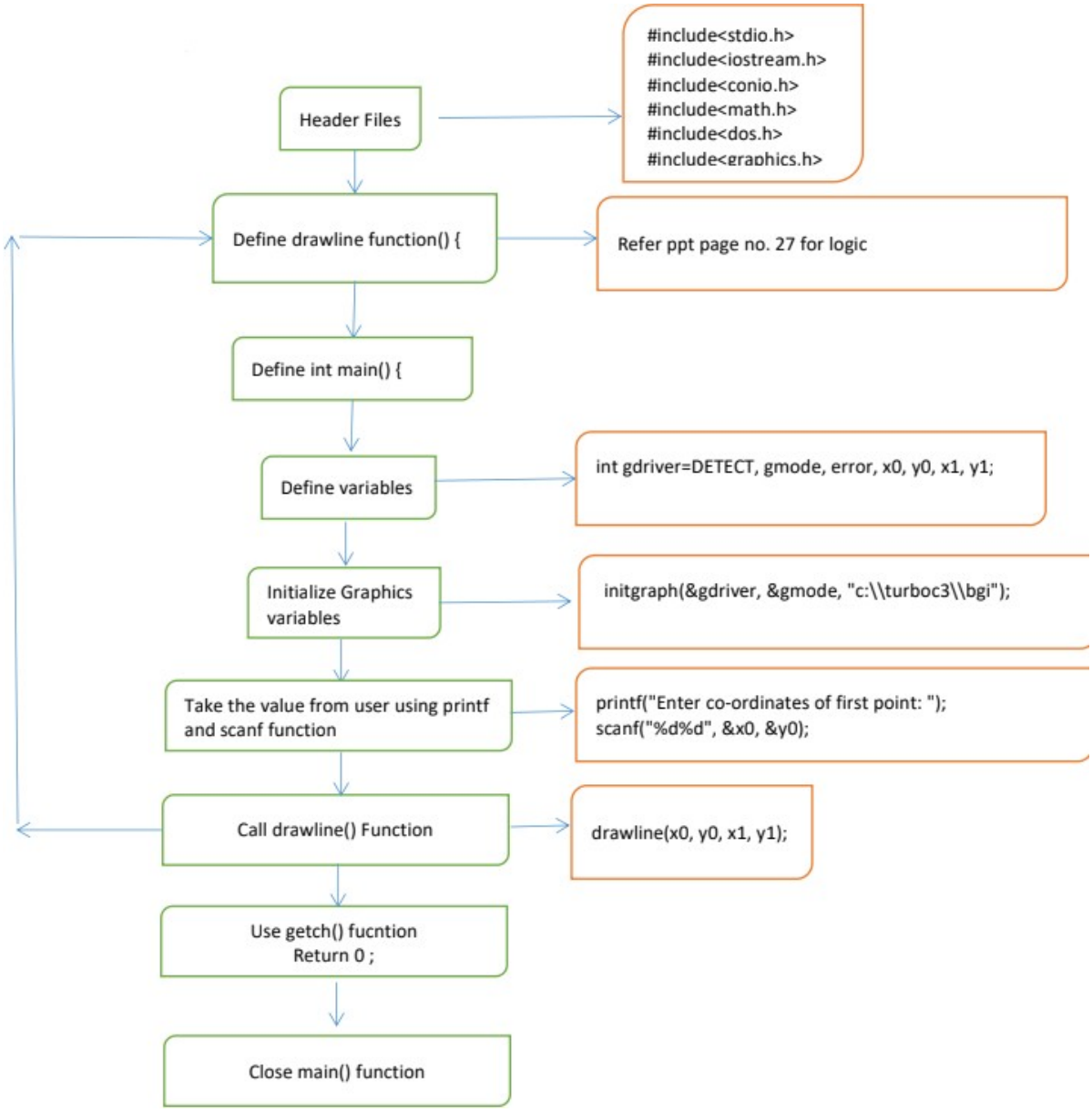
TOPIC:-EXPERIMENT NO:-3

EXPERIMENT:3

Aim : Write a program in c/ c++ to implement Bresenham's line drawing algorithm.

Software used : Turbo C++

Flow chart :



i) To draw a dotted line.

Code:

```
#include <stdio.h>
#include <graphics.h>
#include <math.h>
void main()
{
   int gd = DETECT, gm, x, y, x1, y1, x2, y2, dx, dy, i, e;float
   xinc, yinc;
   initgraph(&gd, &gm, "C:\\TURBOC3\\BGI");
   printf("Enter the starting coordinate of line\n");
   scanf("%d%d", &x1, &y1);
   printf("Enter the ending coordinates of line\n");
   scanf("%d%d", &x2, &y2);
   dx = x2 - x1;
   dy = y2 - y1;
   if (x1 < x2)
      xinc = 1;
   else
      xinc = -1;
   if (y1 < y2)
      yinc = 1;
   else
      yinc = -1;
   x = x1;
   y = y1;
   if (dx >= dy)
   {
      e = (2 * dy) - dx;
      while (x != x2)
      {
         if (e < 0)
            e = e + (2 * dy);
         else
         {
            e = e + (2 * (dy - dx));
```

```
            y = y + yinc;
            y = y + yinc;
        }
        x = x + xinc;
        x = x + xinc;
        putpixel(x, y, WHITE);
    }
}
else
{
    e = (2 * dx) - dy;
    while (y != y2)
    {
        if (e < 0)
            e = e + (2 * dx);
        else
        {
            e = e + (2 * (dx - dy));x
            = x + xinc;
            x = x + xinc;
        }
        y = y + yinc;
        y = y + yinc;
        putpixel(x, y, WHITE);
    }
}
getch();
closegraph();
}
```

OUTPUT:

CONCLUSION: In this experiment we have successfully generated a DOTTED Line using Bresenham's line algorithm.

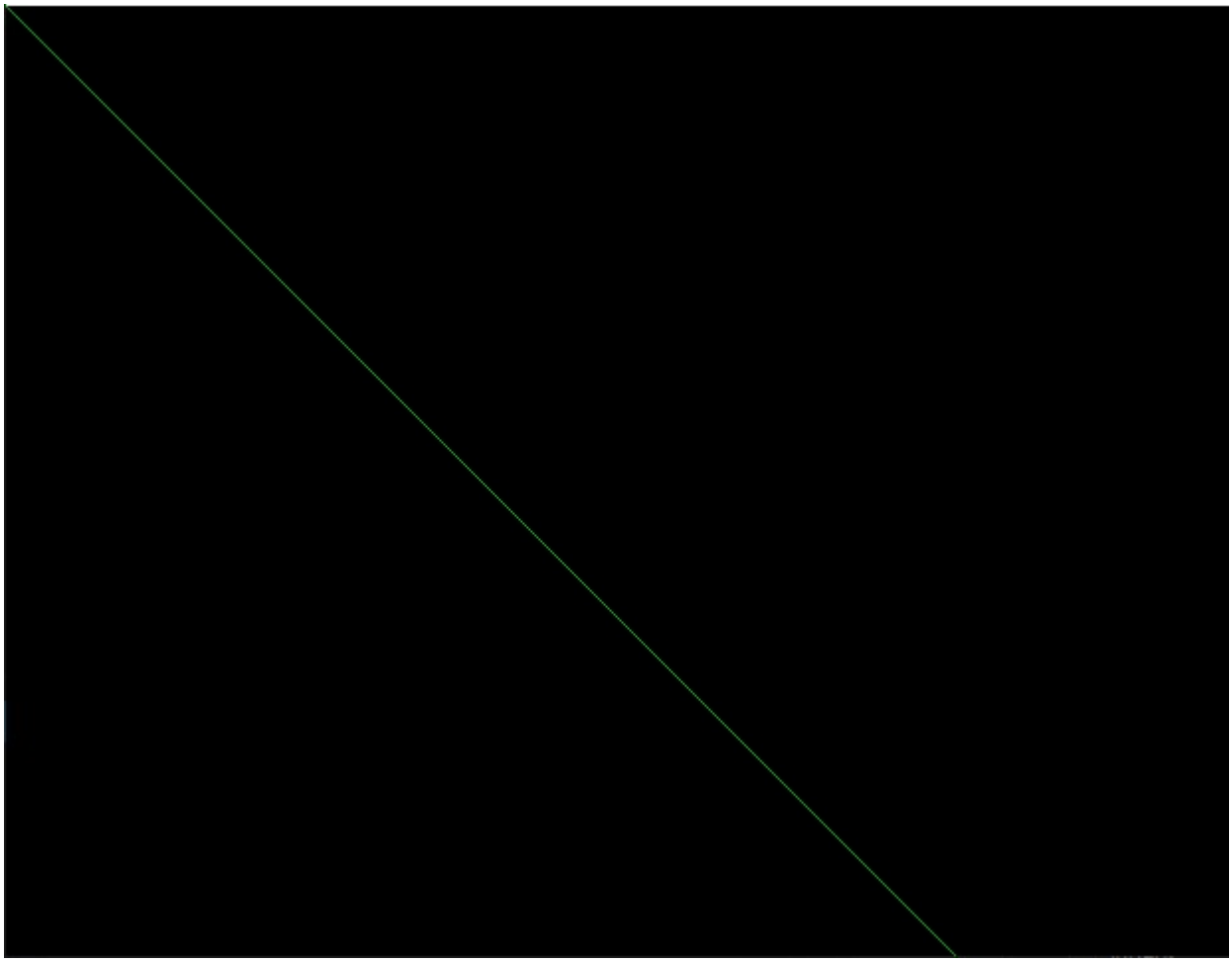
ii) To draw a thin line

Code:

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

int main(void)
{
   int gdriver = DETECT, gmode;
   int i, x, y, x1, y1, x2, y2, dx, dy, length, P0; printf("Enter
   the starting coordinate of line\n");scanf("%d%d", &x1,
   &y1);
   printf("Enter the ending coordinates of line\n");
   scanf("%d%d", &x2, &y2);
   initgraph(&gdriver, &gmode, "C:\\TURBOC3\\BGI");
   dx = abs(x2 - x1);
   dy = abs(y2 - y1);
   if (dx > dy)
      length = dx;
   else
      length = dy;
   putpixel(x1, y1, 2);
   x = x1;
   y = y1;
   P0 = 2 * dy - dx;
   for (i = 0; i % 6 < 4; i++)
      for (i = 0; i <= length; i++)
      {
         if (P0 < 0)
         {
            x = x + 5;
```

```
            y = y + 5;
            putpixel(x, y, 2);
            P0 = P0 + 2 * y;
        }
        else
        {
            x = x + 5;
            y = y + 5;
            putpixel(x, y, 2);
            P0 = P0 + 2 * y - 2 * dx;
        }
    }
  getch();
  closegraph();
  return 0;
}
```

OUTPUT:



CONCLUSION: In this experiment we have successfully generated a THIN Line using Bresenham's line algorithm.

NAME:- SINGH SUDHAM DHARMENDRA

BRANCH:- CSE(AIML)

ROLL NO.:- AIMLD50

SUBJECT:- COMPUTER GRAPHICS

TOPIC:-EXPERIMENT NO:-4

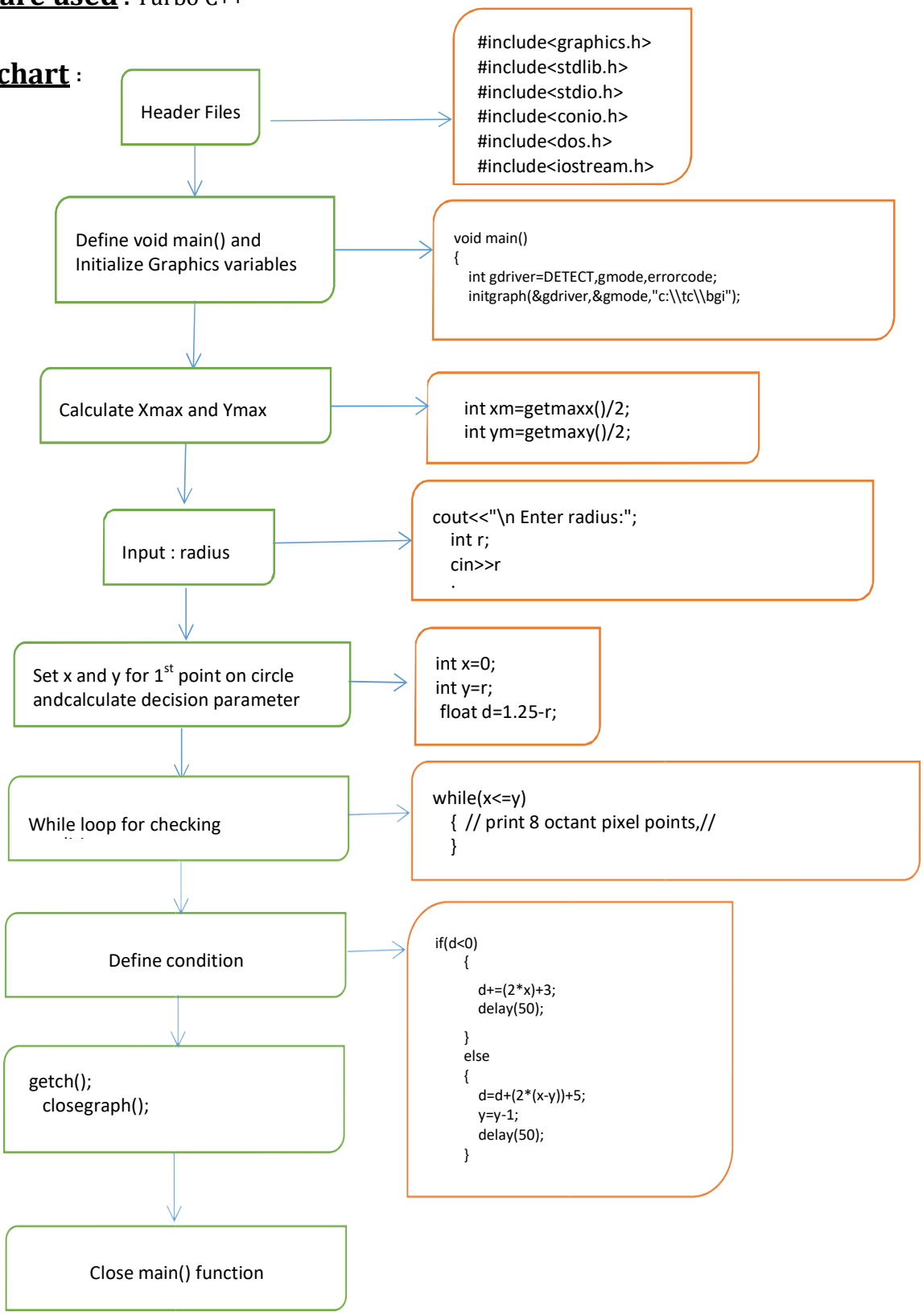
EXPERIMENT : 4

Aim :

Write a program in c to implement Mid-Point circle generating algorithm

Software used : Turbo C++

Flow chart :



Source Code:

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

void main()
{
 int gdriver = DETECT, gmode,errorcode;
 initgraph(&gdriver, &gmode,"c:\\TURBOC3\\BGI");
 int xm = getmaxx()/2;
 int ym = getmaxx()/2;
 int r;
 printf("enter radius\n");
 scanf("%d",&r)
 int x = 0;
 int y = r;
 float d = 1.25 -r;
 while(x<=y)
 {
     putpixel(xm+x,ym-y,15);
     putpixel(xm+y,ym-x,15);
     putpixel(xm+y,ym+x,15);
     putpixel(xm+x,ym+y,15);
     putpixel(xm-x,ym+y,15);
     putpixel(xm-y,ym+x,15);
     putpixel(xm-y,ym-x,15);
     putpixel(xm-x,ym-y,15);
     if(d<0)
     {
        d+=(2*x)+3;
        delay(50);
     }
     else
     {
        d=d+(2*(x-y))+5;
        y=y-1;
        delay(50);
     }
     x=x+1;
 }
```

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

Output:

Conclusion:

From this experiment we have learnt to draw a circle using mid – point algorithm.

NAME: SINGH SUDHAM DHARMENDRA

ROLL NO.: AIMLD50
BRANCH: CSE (AI & ML)
SUBJECT: COMPUTER GRAPHICS
TOPIC: EXPERIMENT NO. 5

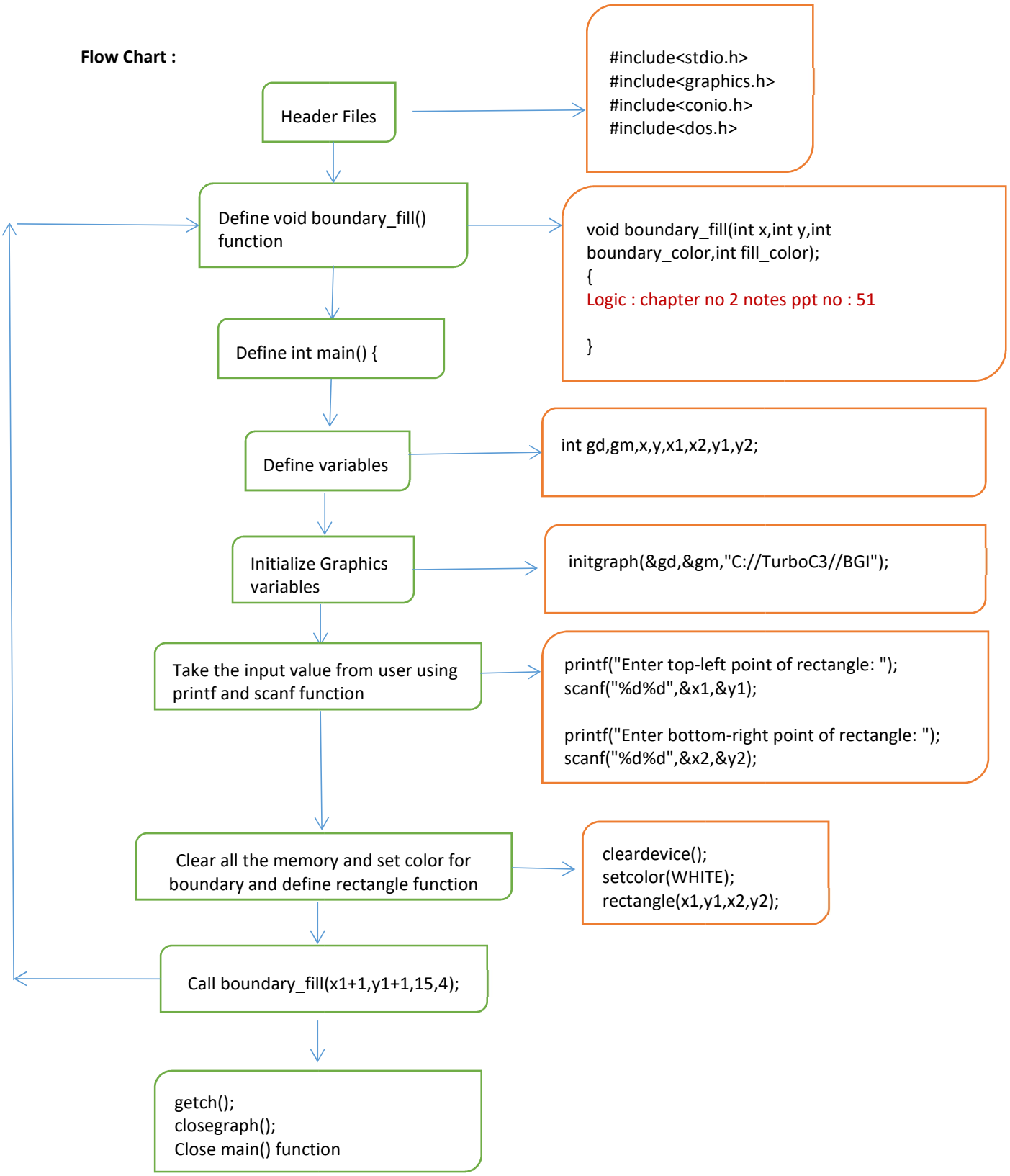
Experiment No: 5

Aim: Write a program in c to implement

i) Boundary Fill algorithm ii) Flood Fill algorithm.

Software used : Turbo C++

Flow Chart :

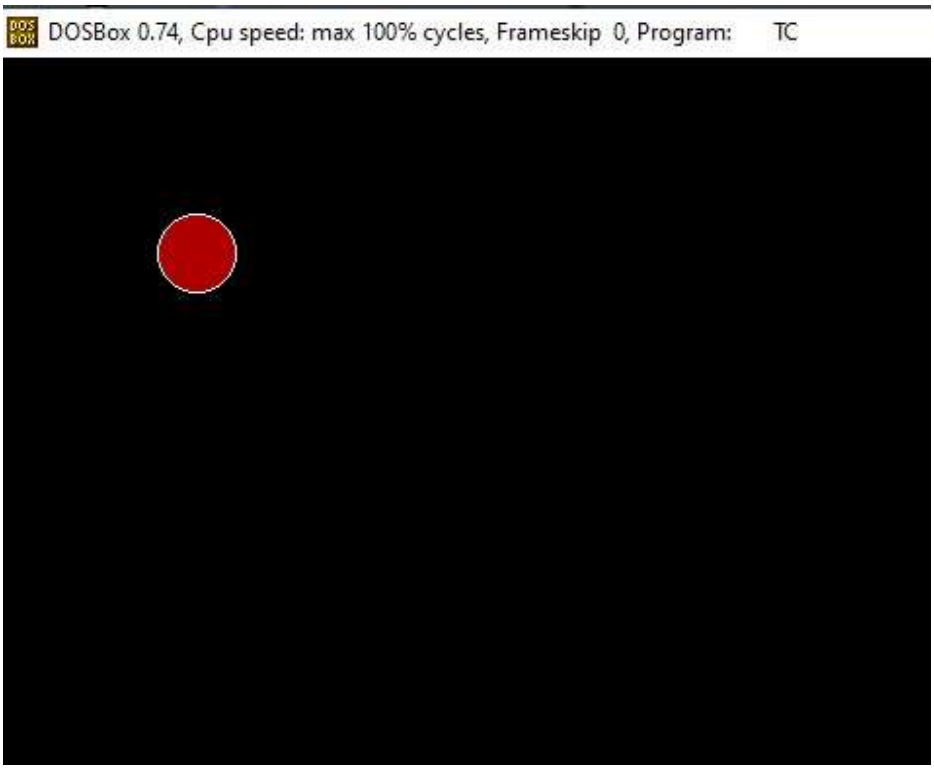


1) Boundary Fill algorithm

Source Code:

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

void boundaryfill(int x,int y,int f_color,int b_color)
{ if(getpixel(x,y)!=b_color && getpixel(x,y)!=f_color)
   { putpixel(x,y,f_color);
      boundaryfill(x+1, y,
      f_color,b_color); boundaryfill(x,
      y+1, f_color,b_color);
      boundaryfill(x-1, y, f_color,b_color);
      boundaryfill(x, y-1, f_color,b_color);
   }
} int
main()
{
   int gm,gd=DETECT,radius;
   int x,y;

   printf("Enter x and y positions for circle\n");
   scanf("%d%d",&x,&y);
   printf("Enter radius of circle\n");
   scanf("%d",&radius);

   initgraph(&gd,&gm,"C:\\TURBOC3\\BGI");
   circle(x,y,radius); boundaryfill(x,y,4,15);
   delay(5000); closegraph();

   return 0;
}
```

Output:



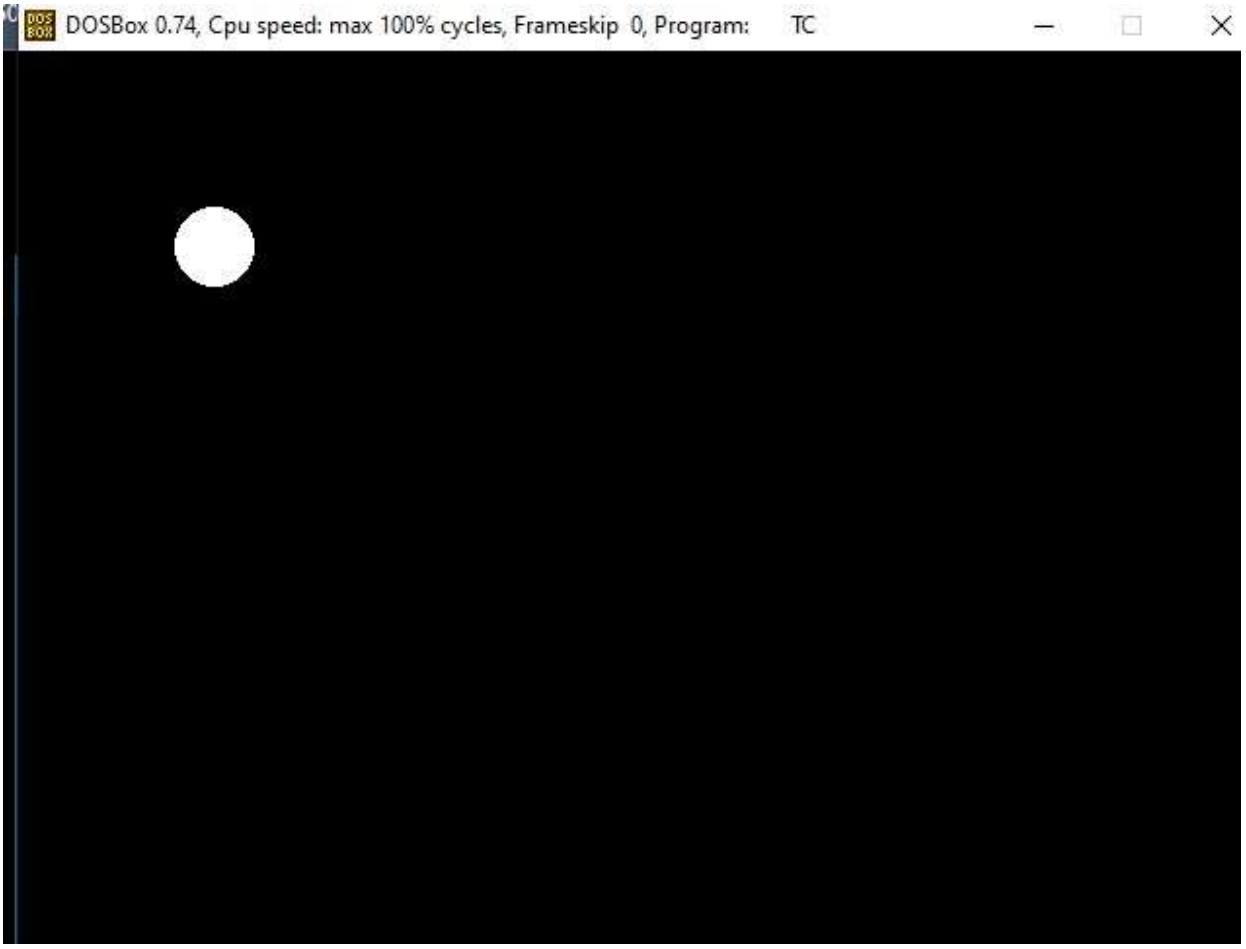
Conclusion: Here's a circle obtained by using the BOUNDARY FILL algorithm.

2) Flood Fill algorithm

Source Code:

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

void floodFill(int x,int y,int oldcolor, int newcolor)
{ if(getpixel(x,y) == oldcolor)
        {
                putpixel(x,y,newcolor);
                floodFill(x+1,y,oldcolor,newcolor);
                floodFill(x,y+1,oldcolor,newcolor); floodFill(x-
                1,y,oldcolor,newcolor);
        floodFill(x,y-1,oldcolor,newcolor);
        }
}
//getpixel(x,y) gives the color of specified pixel

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

        printf("Enter x and y positions for circle\n");
        scanf("%d%d",&x,&y);
        printf("Enter radius of circle\n");
        scanf("%d",&radius);

        initgraph(&gd,&gm,"C:\\TURBOC3\\BGI")
        ; circle(x,y,radius); floodFill(x,y,0,15);

        delay(5000);
        closegraph();
        return 0 ;
}
```

Output:



Conclusion: Here's a circle obtained by using the FLOOD FILL algorithm.

NAME:- SINGH SUDHAM DHARMENDRA

BRANCH:- CSE(AIML)

ROLL NO.:- AIMLD50

SUBJECT:- COMPUTER GRAPHICS

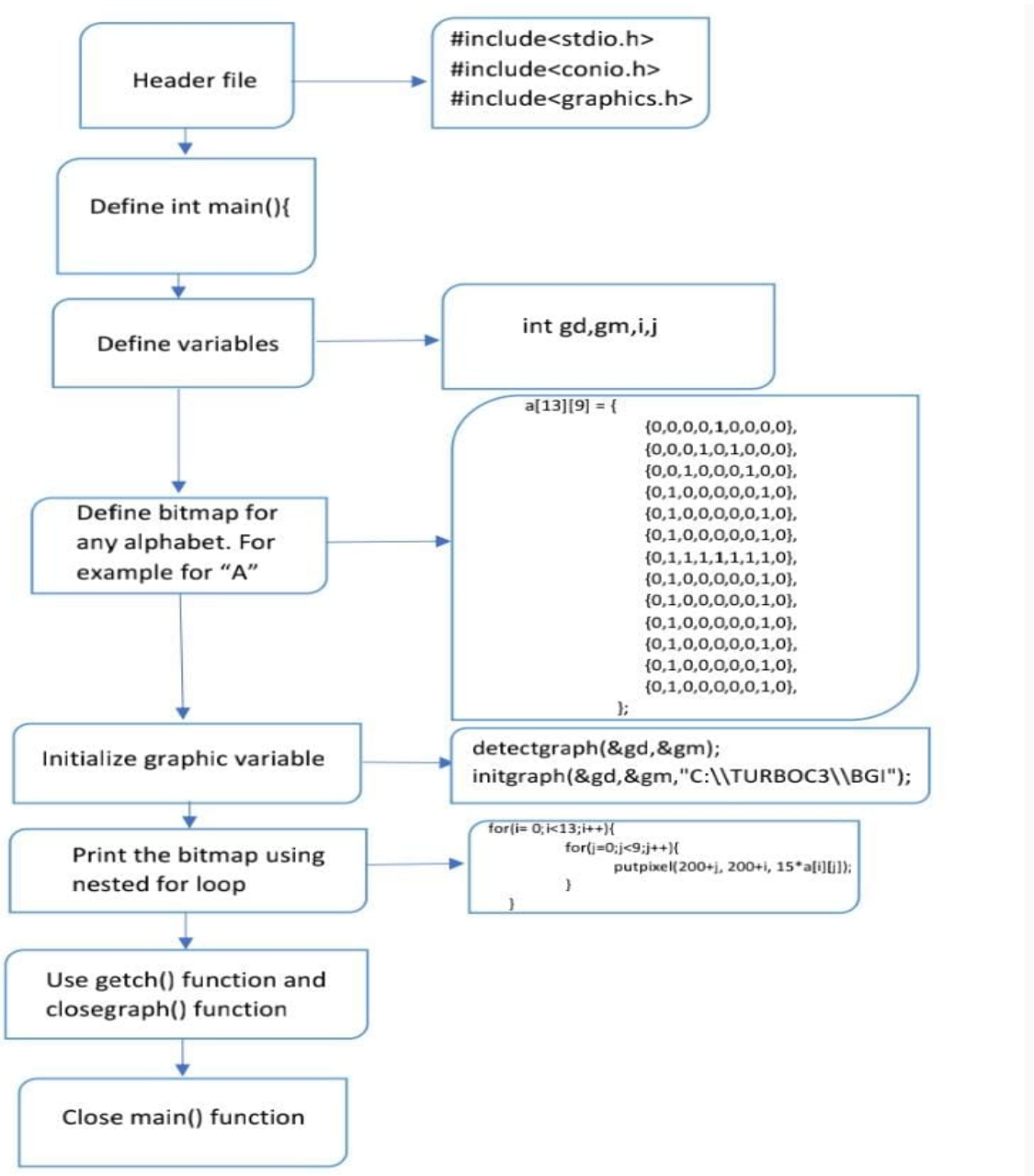
TOPIC:-EXPERIMENT NO:-6

EXPERIMENT NO. 6

AIM: Write a program in c for character generation -bitmap method.

SOFTWARE USED: Turbo C++

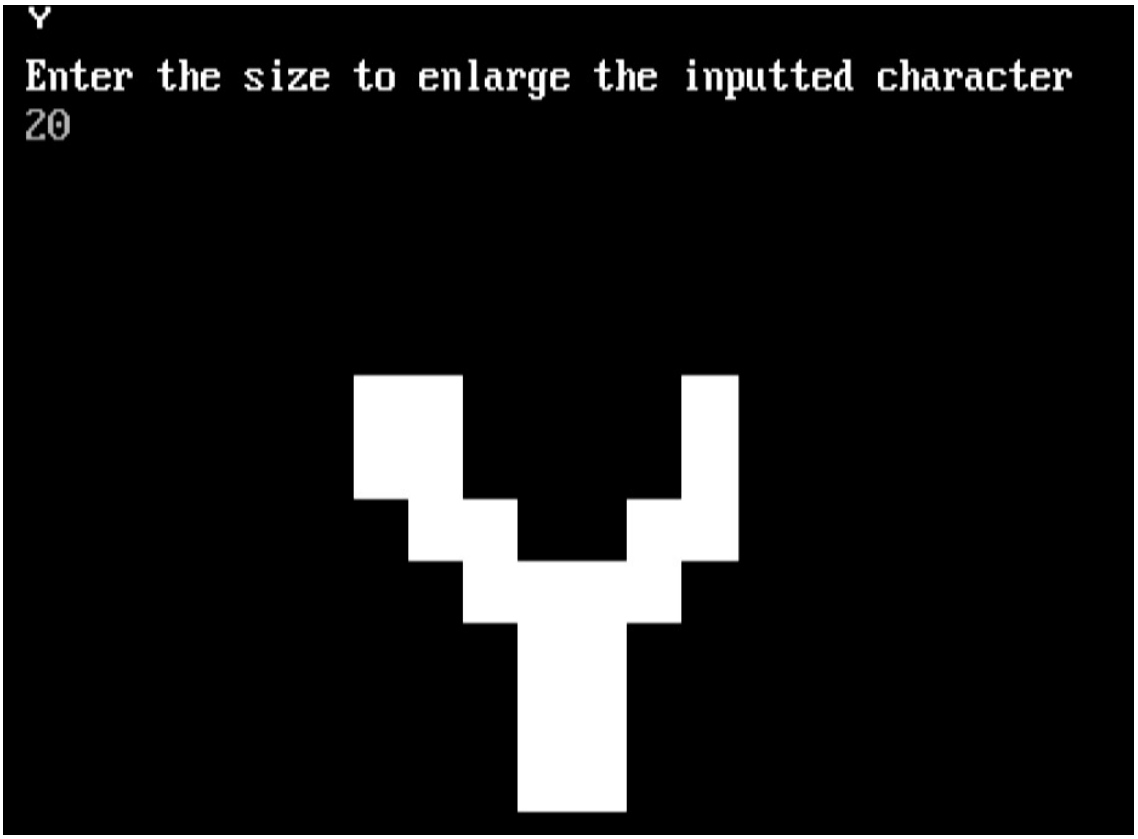
FLOWCHART:



SOURCE CODE:

```
#include<conio.h>
#include<stdio.h>
#include<graphics.h>
void main()
{
int gd= DETECT, gm,i,j,m;
char c[8][8],s[1];  // needed to delcare 's' as string inorder to use it with
outtextxy()
initgraph(&gd,&gm,"..\\bgi");
printf("Enter a Character:");
scanf("%s",s);
clearviewport();
outtextxy(1,1,s);
for(i=0;i<textwidth("S");i++)      // any  capital character can be given
instead of S
{
for(j=0;j<textheight("S");j++)
{
c[i][j]=getpixel(i,j);
}
}
printf("Enter the size to enlarge the inputted character \n");
scanf("%d",&m);
for(i=0;i<8;i++)
{
for(j=0;j<8;j++)
{
setfillstyle(SOLID_FILL,c[i][j]);
bar(100+(i*m),100+(j*m),100+(i+1)*m,100+(j+1)*m);
}
}
getch();
closegraph();
}
```

OUTPUT:



CONCLUSION:

From this experiment we successfully generated a character using Bitmap method.

NAME:- SINGH SUDHAM DHARMENDRA

BRANCH:- CSE(AIML)

ROLL NO.:- AIMLD50

SUBJECT:- COMPUTER GRAPHICS

TOPIC:-EXPERIMENT NO:-7

Experiment No : 7

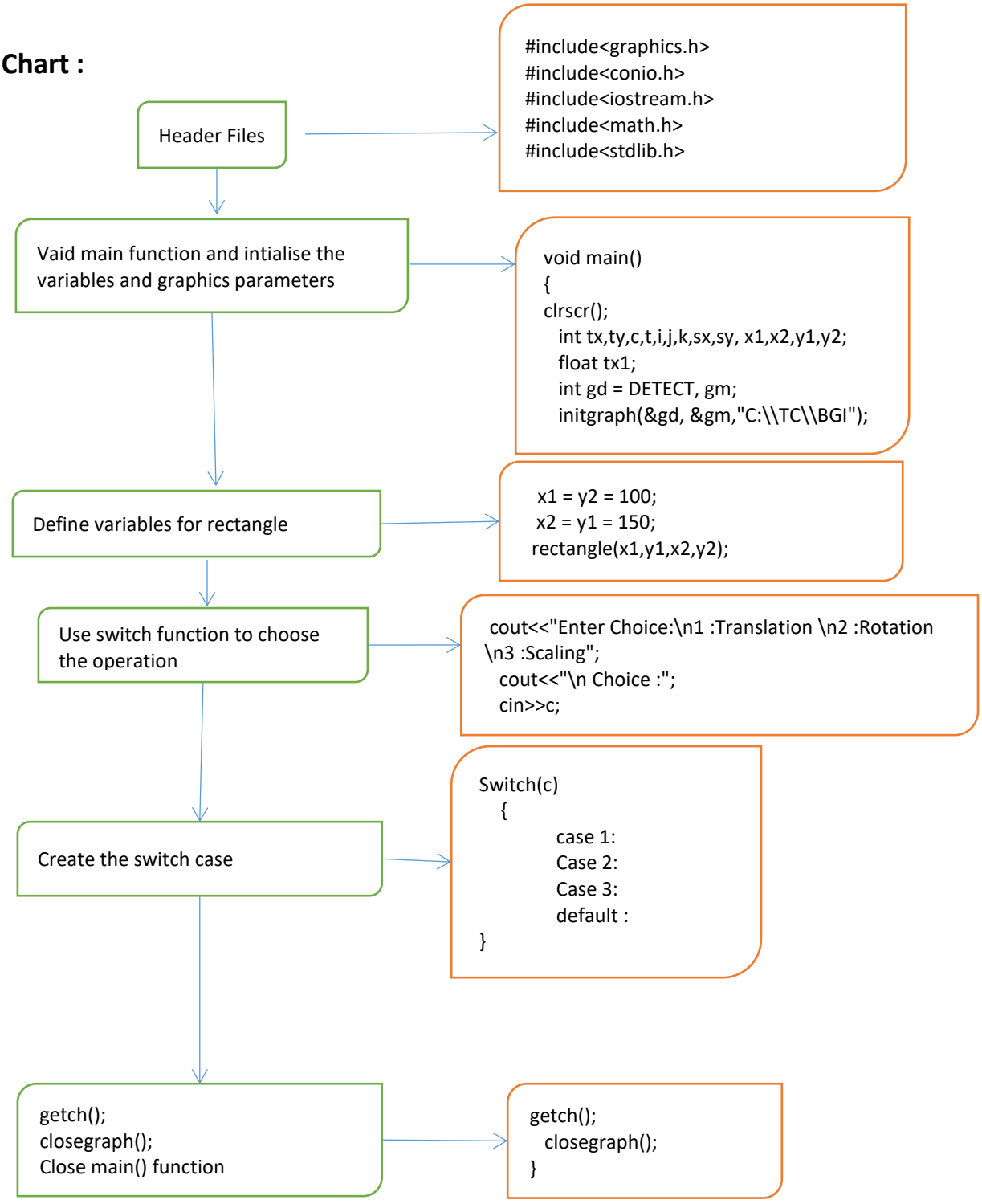
Aim :

Write a program in c to implement 2D transformations

i) Translation ii) Rotation iii) Scaling

Software used : Turbo C++

Flow Chart :

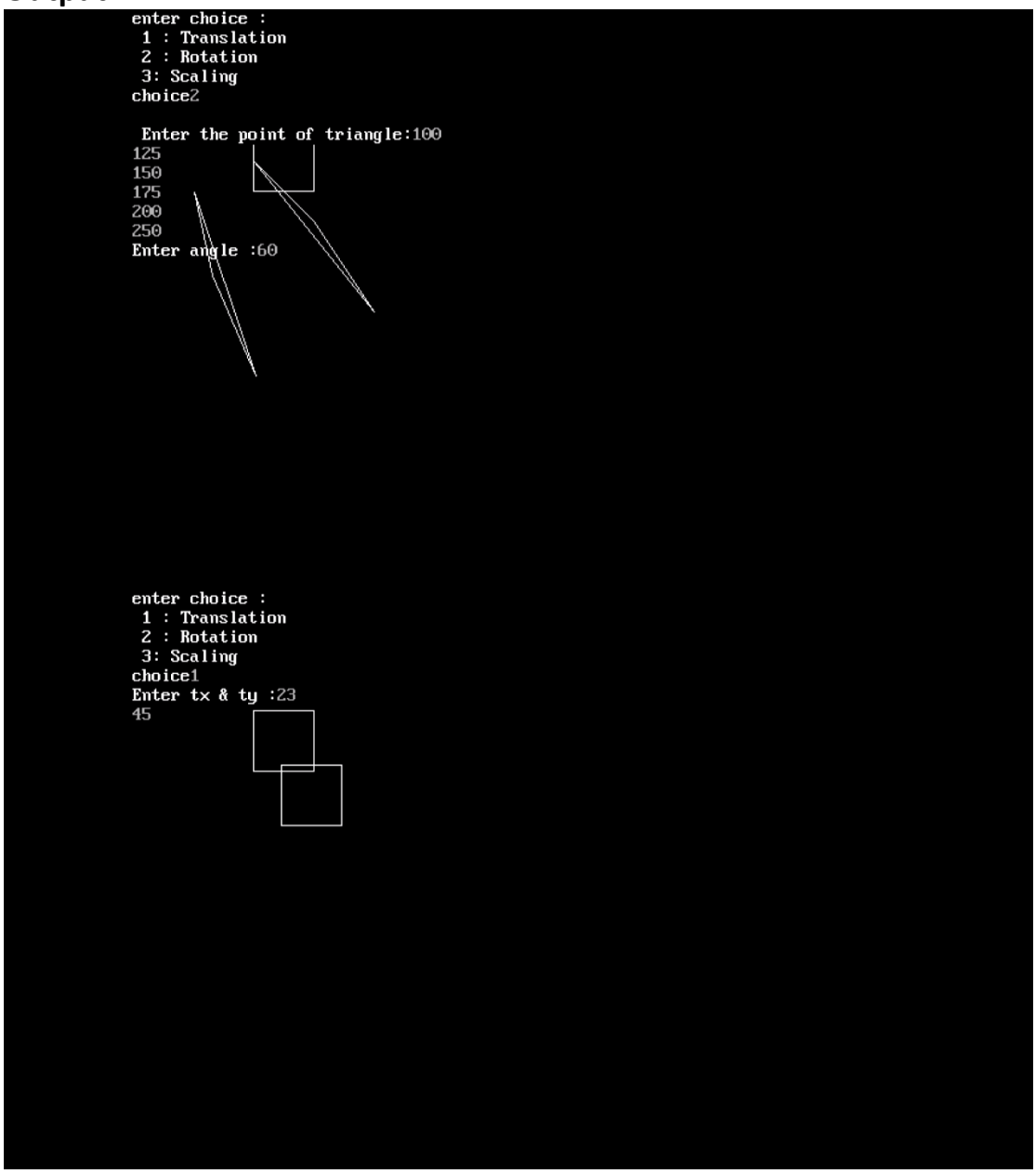


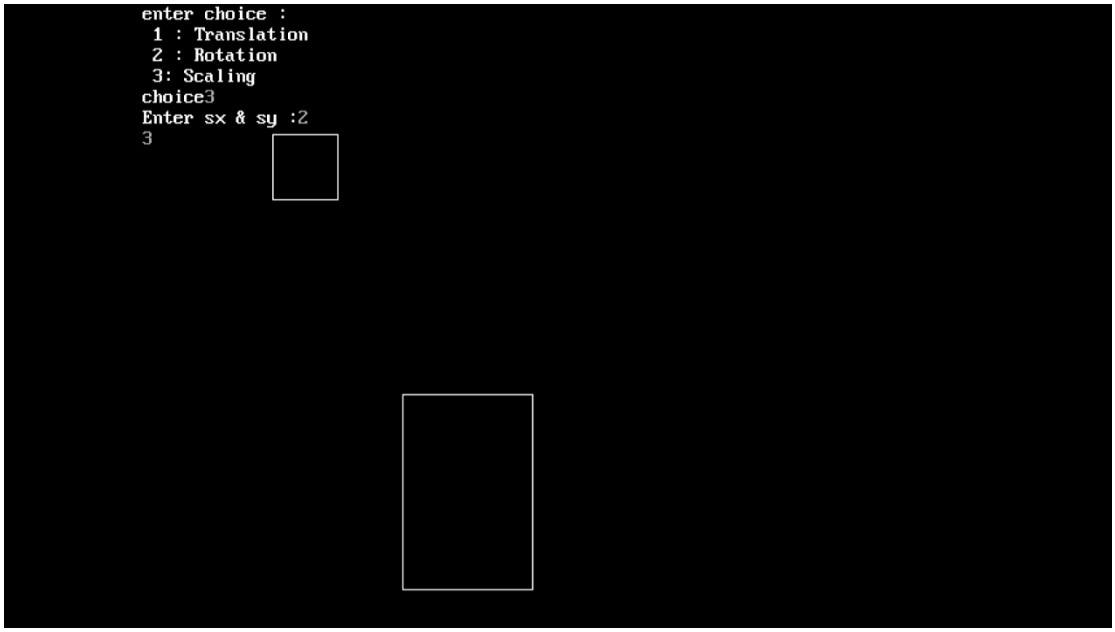
Source Code:

```
#include<stdio.h>
#include<graphics.h>
#include<math.h>
#include<stdlib.h>
#include<dos.h>

int main()
{
   int tx,ty,c,t,i,j,k,sx,sy,x1,x2,y1,y2,t1,t2,t3,t4X1,Y1,X2,Y2,X3,Y3,rrx,nx1,ny1,nx2,ny2,nx3,ny3;
   float tx1;
   int gd = DETECT, gm;
   initgraph(&gd,&gm,"C://Turboc3//BGI");
   x1 = y2 = 100;
   x2 = y1 = 150;
   rectangle(x1,y1,x2,y2);
   printf("enter choice :\n 1 : Translation\n 2 : Rotation\n 3: Scaling");
   printf("\nchoice");
   scanf("%d",&c);
   switch(c){
          case '1':
            printf("Enter tx & ty :");
            scanf("%d%d",&tx,&ty);
            t1=x1+tx;
            t2=y1+ty;
            t3=x2+tx;
            t4=y2+ty;
            rectangle(t1,t2,t3,t4);
            break;
     case '2' :
                                printf(" \n Enter the point of triangle:");
                                scanf("%d%d%d%d%d%d",&X1,&Y1,&X2,&Y2,&X3,&Y3);
                                line(X1,Y1,X2,Y2);
                                line(X2,Y2,X3,Y3);
                                line(X3,Y3,X1,Y1);
                                int r;
                                printf("Enter angle :");
                                scanf("%d",&r);
                                rx=r*(3.14/180);

                                nx1 =abs( X1* cos(rx) - Y1*sin(rx));
                                ny1 =abs( Y1* cos(rx) + X1*sin(rx));

                                nx2 = abs( X2* cos(rx) -Y2*sin(rx));
                                ny2 = abs( Y2* cos(rx) + X2*sin(rx));

                                nx3 = abs (X3* cos(rx) - Y3*sin(rx));
                                ny3 = abs (Y3* cos(rx) + X3*sin(rx));

                                line(nx1,ny1,nx2,ny2);
                                line(nx2,ny2,nx3,ny3);
                                line(nx3,ny3,nx1,ny1);

                                break;
     case '3' :
                        printf("Enter sx & sy :");
```

```
                    scanf("%d%d",&sx,&sy);
                    rectangle(x1*sx, y1*sy, x2*sx, y2*sy);
                    break;
    default :
                    printf("Not a valid choice");
  }
  getch();
  closegraph();
  return 0;
}
```

Output:





Conclusion :

From this experiment we have learned the algorithm for scaling, translating and rotating a 2-Dimensional object.

NAME: SINGH SUDHAM DHARMENDRA

ROLL NO.: AIMLD50

BRANCH: CSE (AI & ML)

SUBJECT: COMPUTER GRAPHICS

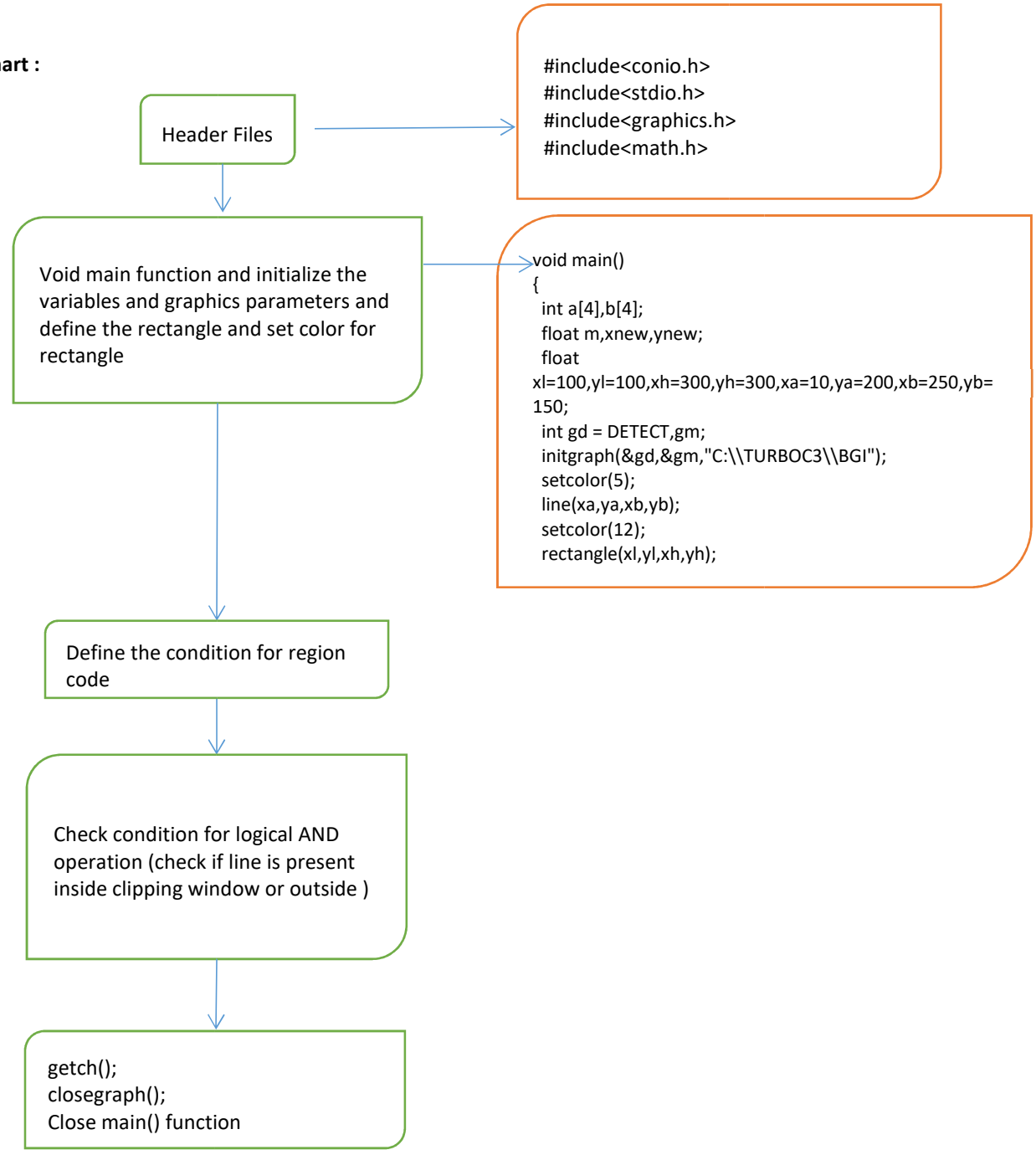
TOPIC: EXPERIMENT NO. 8

Experiment No: 8

Aim: Write a program in c to implement Line clipping algorithm Cohen Sutherland

Software used : Turbo C++

Flow Chart :



SOURCE CODE:

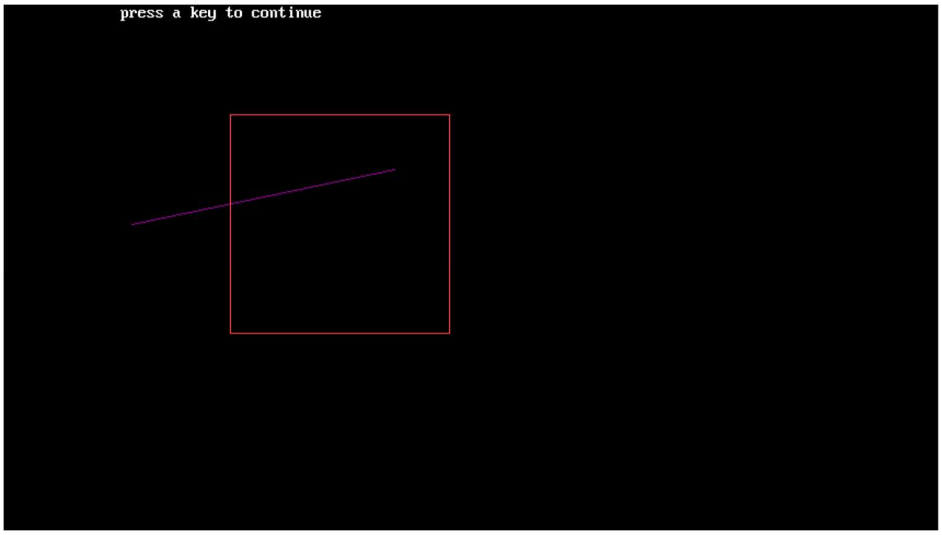
```
#include<conio.h>
#include<stdio.h>
#include<graphics.h>
#include<math.h>

void main()
{
  int a[4],b[4];
  float m,xnew,ynew;
  float xl=100,yl=100,xh=300,yh=300,xa=10,ya=200,xb=250,yb=150;
  int gd = DETECT,gm;
  initgraph(&gd,&gm,"C:\\TURBOC3\\BGI");
  setcolor(5);
  line(xa,ya,xb,yb);
  setcolor(12);
  rectangle(xl,yl,xh,yh);
  m = (yb-ya)/(xb-xa);

  if(xa < xl)
    a[3] = 1;
  else a[3] = 0;

  if(xa>xh)
   a[2] = 1;
  else a[2] = 0;

  if(ya < yl)
   a[1] = 1;
  else a[1] = 0;

  if (ya > yh)
```

```
 a[0] = 1;
 else a[0] = 0;

if(xb < xl)
  b[3] = 1;
else b[3] = 0;

if(xb>xh)
 b[2] = 1;
else b[2] = 0;

if(yb < yl)
 b[1] = 1;
else b[1] = 0;

if (yb > yh)
 b[0] = 1;
 else b[0] = 0;

printf("press a key to continue");
getch();
if(a[0] == 0 && a[1] == 0 && a[2] == 0 && a[3] == 0 && b[0] == 0 &&
b[1] == 0 && b[2] == 0 && b[3] == 0 )
{

 printf("no clipping");
 line(xa,ya,xb,yb);
}
else if(a[0]&&b[0] || a[1]&&b[1] || a[2]&&b[2] || a[3]&&b[3])
{
 clrscr();
 printf("line discarded");
 rectangle(xl,yl,xh,yh);
}
```

```
else
{
 if(a[3] == 1 && b[3]==0)
 {
  ynew  = (m * (xl-xa)) + ya;
  setcolor(12);
  rectangle(xl,yl,xh,yh);
  setcolor(0);
  line(xa,ya,xb,yb);
  setcolor(15);
  line(xl,ynew,xb,yb);
 }
 else if(a[2] == 1 && b[2] == 0)
 {
  ynew  = (m * (xh-xa)) + ya;
  setcolor(12);
  rectangle(xl,yl,xh,yh);
  setcolor(0);
  line(xa,ya,xb,yb);
  setcolor(15);
  line(xl,ynew,xb,yb);
 }
 else if(a[1] == 1 && b[1] == 0)
 {
  xnew = xa + (yl-ya)/m;
  setcolor(0);
  line(xa,ya,xb,yb);
  setcolor(15);
  line(xnew,yh,xb,yb);
 }

 else if(a[0] == 1 && b[0] == 0)
 {
  xnew = xa + (yh-ya)/m;
```

```
    setcolor(0);
    line(xa,ya,xb,yb);
    setcolor(15);
    line(xnew,yh,xb,yb);
   }
  }
  getch();
  closegraph();
}
```

OUTPUT:



CONCLUSION: IN THIS EXPERIMENT WE HAVE SUCCESSFULLY IMPLEMENTED LINE CLIPPING USING Cohen Sutherland ALGORITHM.

NAME: SINGH SUDHAM DHARMENDRA

ROLL NO.: AIMLD50

BRANCH: CSE (AI & ML)

SUBJECT: COMPUTER GRAPHICS

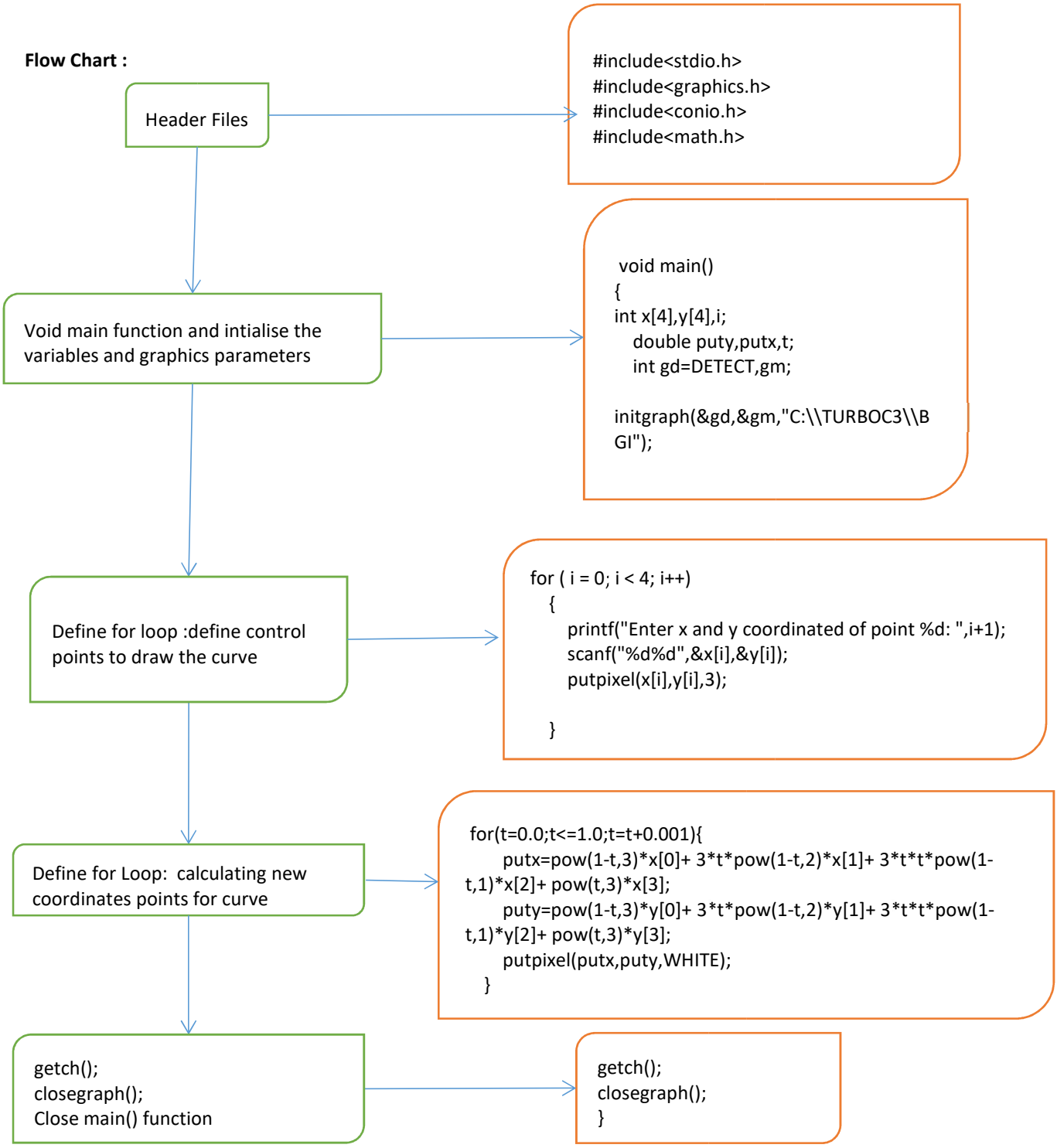
TOPIC: EXPERIMENT NO. 9

Experiment No: 9

Aim: Write a program in c to implement Bezier curve.

Software used : Turbo C++

Flow Chart :

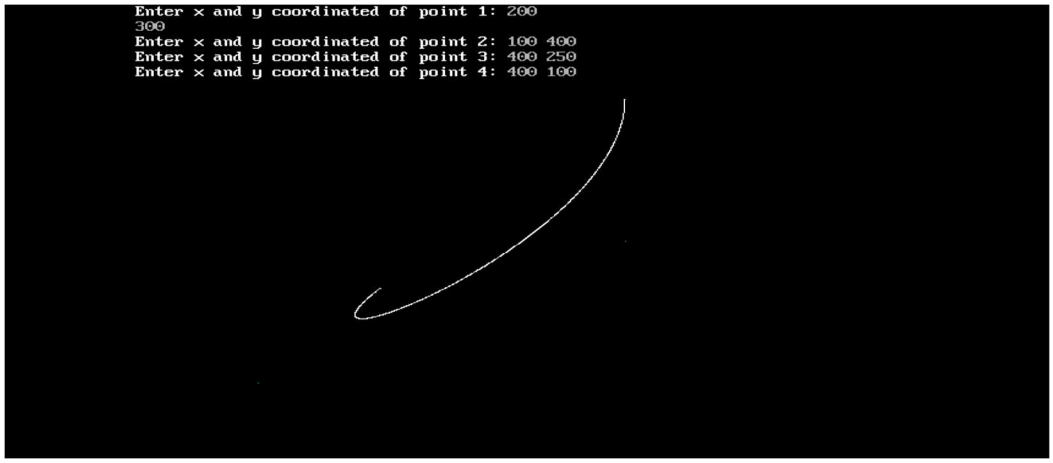


Source Code:

```
#include<stdio.h>

#include<graphics.h>

#include<conio.h>

#include<math.h>

void main(){

   int x[4],y[4],i;

   double puty,putx,t;

   int gd=DETECT,gm;

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

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

   {

      printf("Enter x and y coordinated of point %d:
",i+1);

      scanf("%d%d",&x[i],&y[i]);

      putpixel(x[i],y[i],3);



   }
```

```
   for(t=0.0;t<=1.0;t=t+0.001){

      putx=pow(1-t,3)*x[0]+ 3*t*pow(1-t,2)*x[1]+
3*t*t*pow(1-t,1)*x[2]+ pow(t,3)*x[3];

      puty=pow(1-t,3)*y[0]+ 3*t*pow(1-t,2)*y[1]+
3*t*t*pow(1-t,1)*y[2]+ pow(t,3)*y[3];

      putpixel(putx,puty,WHITE);

   }

   getch();

   closegraph();



}
```

OUTPUT:



CONCLUSION:

IN THIS EXPERIMENT WE HAVE SUCCESSFULLY IMPLEMENTED BEZIER CURVE USING TURBO C++.

NAME: SINGH SUDHAM DHARMENDRA

ROLL NO.: AIMLD50

BRANCH: CSE (AI & ML)

SUBJECT: COMPUTER GRAPHICS

EXPERIMENT NO. 10

MINI PROJECT

MINI PROJECT TITLE: VISUALIZATION OF SIGNAL TRANSMISSION

GROUP MEMBERS: 1. RUPARELIYA AKSHAR JAYANTI (41)

2. SINGH SUDHAM DHARMENDRA (50)

3. ANSARI HANZALA MOHD IMAMUDDIN (03)

4. CHIKANKAR PRATHAMESH SHIVAJI (08)

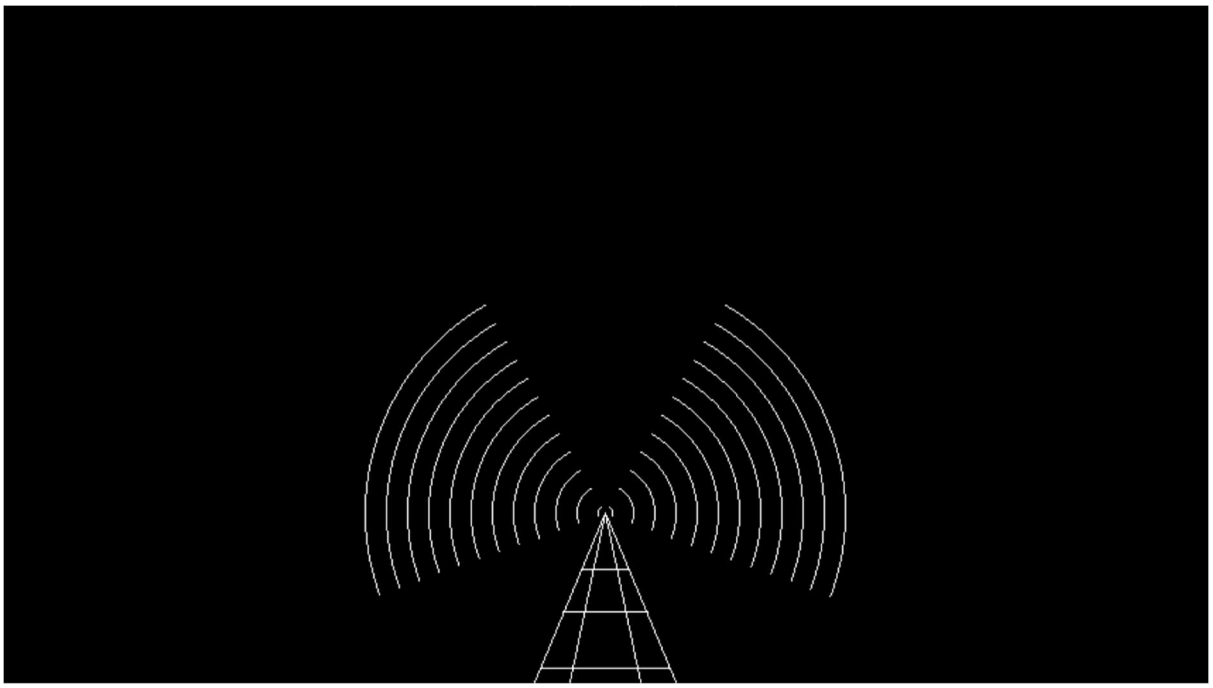
SOFTWARE USED: TURBO C++

SOURCE CODE:

```
#include <stdio.h>
#include <conio.h>
#include <graphics.h>
#include <dos.h>
int main()
{
    int gdriver = DETECT, gmode, err;
    int midx, y, radius = 5;
    int k = 0, stangle, endangle;

    initgraph(&gdriver, &gmode, "C:\\TURBOC3\\BGI");

    err = graphresult();
    if (err != grOk)
    {   /* error occurred */
        printf("Graphics Error: %s\n",grapherrormsg(err));
        getch();
        return 0;
    }
    /* mid position in x-axis */
    midx = getmaxx() / 2;

    /* calculating the position of y */
    y = (3 * getmaxy()) / 4;

    /* start and end angles of signals */
    stangle = 120;
    endangle = 200;

    while (!kbhit())
    {
        k = 0, radius = 5;
        /* clears graphic screen */
        cleardevice();

        /* construct antenna using lines */
        line(midx - 50, getmaxy(), midx, y);
        line(midx + 50, getmaxy(), midx, y);
        line(midx - 25, getmaxy(), midx, y);
        line(midx + 25, getmaxy(), midx, y);
        line(midx - 45, getmaxy() - 10, midx + 45, getmaxy() - 10);
        line(midx - 30, getmaxy() - 50, midx + 30, getmaxy() - 50);
        line(midx - 16, getmaxy() - 80, midx + 16, getmaxy() - 80);
        /* signals from the antenna */

        while (k < 18)
        {
            /* signal at the left side */
            arc(midx, y, stangle, endangle, radius);
            /* signal at the right side */
            arc(midx, y, 0, 60, radius);
```

```
            arc(midx, y, 340, 360, radius);
            radius = radius + 15;
            delay(50);
            k++;
        }
    }
    getch();

    /* deallocate memory allocated for graphic screen */
    closegraph();

    return 0;
}
```

OUTPUT:



CONCLUSION:

FROM THE ABOVE PROGRAM AND OUTPUT WE CONCLUDE THAT THE PROJECT IS ABLE TO PERFORM VISUALIZATION OF SIGNAL TRANSMISSION I.E. THE TRANSMISSION OF ELECTROMAGNETIC WAVES FROM AN ANTENNAS OR SATELLITE.

Computer Graphics Mini Project

- GUIDED BY VAIBHAV PALAV

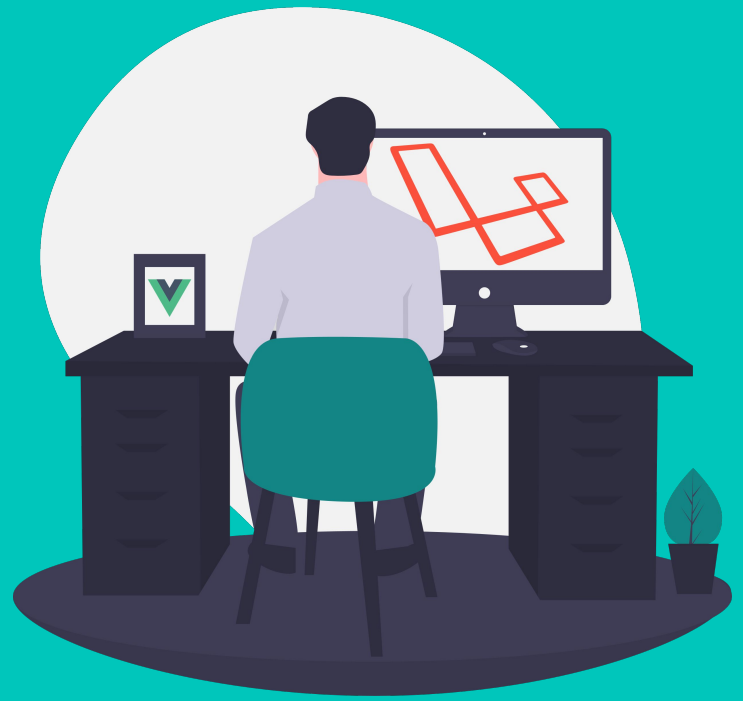
Team Members :

1. AIMLD50_SINGH SUDHAM DHARMENDRA
2. AIMLD03_ANSARI HANZALA MOHD IMAMUDDIN
3. AIMLD41_RUPARELIYA AKSHAR JAYANTI
4. AIMLD08_CHIKANKAR PRATHAMESH SHIVAJI

Visualization Of Signal Transmission

OUTLINE :

1. Introduction
2. Program
3. Output
4. Conclusion



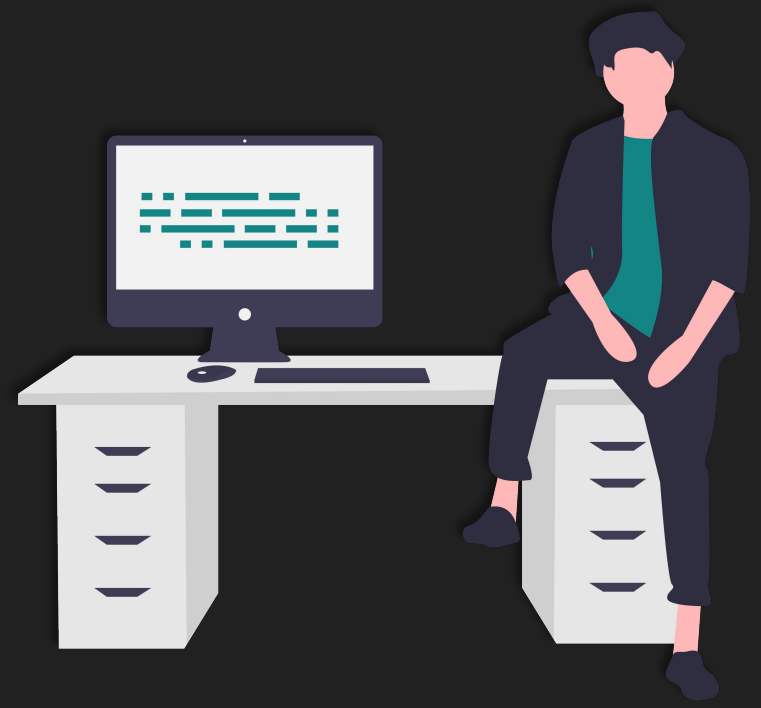
Introduction

1. In this Mini Project, We are going to develop an Animation using Computer Graphics for Visualization of Signal Transmission .
2. Wireless Communication is the fastest growing and most vibrant technological areas in the communication field.
3. Wireless Communication is a method of transmitting information from one point to other, without using any connection like wires, cables or any physical medium.
4. Antennas are electrical devices that transform the electrical signals to radio signals in the form of Electromagnetic (EM) Waves and vice versa.
5. These Electromagnetic Waves propagates through space. Hence, both transmitter and receiver consists of an antenna.

Introduction

Some Built-in Functions of Graphics :

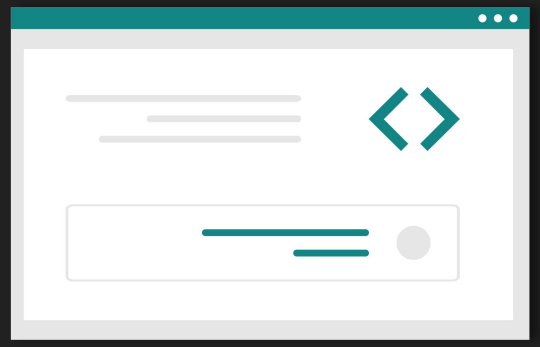
1. **initgraph() :** initgraph initializes the graphics system by loading a graphics driver from disk and putting the system into graphics mode.
2. **graphresult() :** graphresult returns the error code for the last graphics operation that reported an error and resets the error level to grOk.
3. **getmaxx() :** getmaxx returns the maximum x screen coordinate for the current graphics driver and mode. getmaxx is invaluable for centering, determining the boundaries of a region onscreen, and so on.
4. **getmaxy() :** getmaxx returns the maximum x screen coordinate



Program

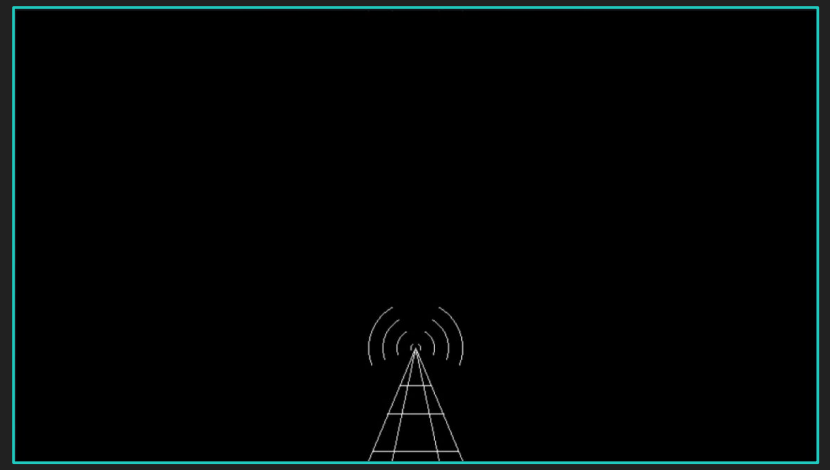
```c
#include <stdio.h>
#include <conio.h>
#include <graphics.h>
#include <dos.h>
int main()
{
    int gdriver = DETECT, gmode, err;
    int midx, y, radius = 5;
    int k = 0, stangle, endangle;

    initgraph(&gdriver, &gmode, "C:\\TURBOC3\\BGI");

    err = graphresult();
    if (err != grOk)
    {   /* error occurred */
        printf("Graphics Error: %s\n",grapherrormsg(err));
        getch();
        return 0;
    }
    /* mid position in x-axis */
    midx = getmaxx() / 2;

    /* calculating the position of y */
    y = (3 * getmaxy()) / 4;

    /* start and end angles of signals */
    stangle = 120;
    endangle = 200;
```

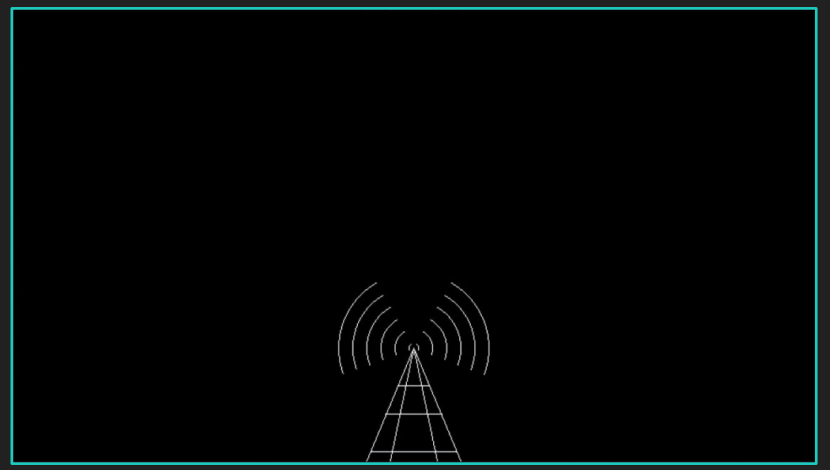
Program

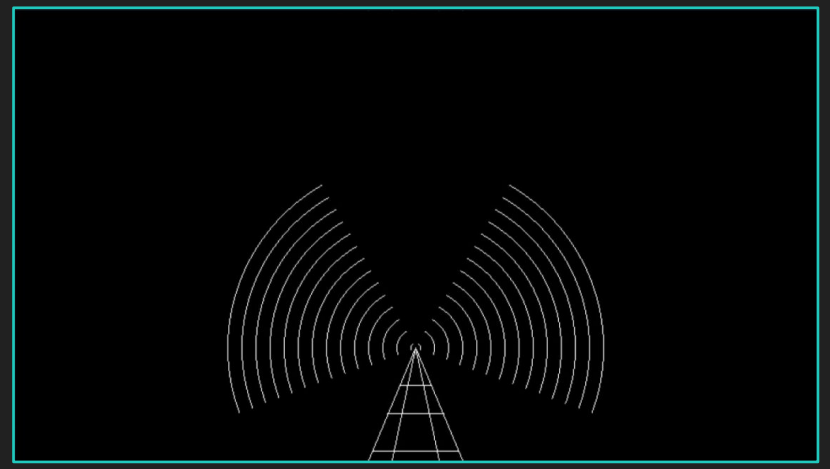


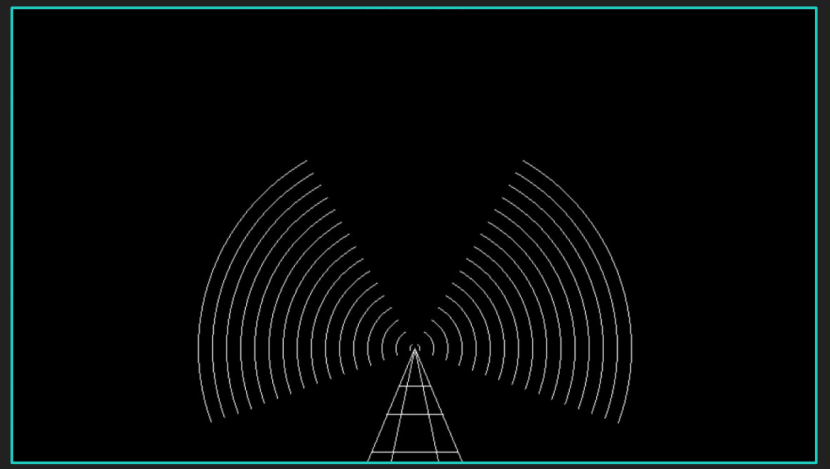
```
    while (!kbhit())
    {
        k = 0, radius = 5;
        /* clears graphic screen */
        cleardevice();

        /* construct antenna using lines */
        line(midx - 50, getmaxy(), midx, y);
        line(midx + 50, getmaxy(), midx, y);
        line(midx - 25, getmaxy(), midx, y);
        line(midx + 25, getmaxy(), midx, y);
        line(midx - 45, getmaxy() - 10, midx + 45, getmaxy() - 10);
        line(midx - 30, getmaxy() - 50, midx + 30, getmaxy() - 50);
        line(midx - 16, getmaxy() - 80, midx + 16, getmaxy() - 80);
        /* signals from the antenna */

        while (k < 18)
        {
            /* signal at the left side */
            arc(midx, y, stangle, endangle, radius);
            /* signal at the right side */
            arc(midx, y, 0, 60, radius);
            arc(midx, y, 340, 360, radius);
            radius = radius + 15;
            delay(50);
            k++;
        }
    }
    getch();

    /* deallocate memory allocated for graphic screen */
    closegraph();

    return 0;
}
```

Output



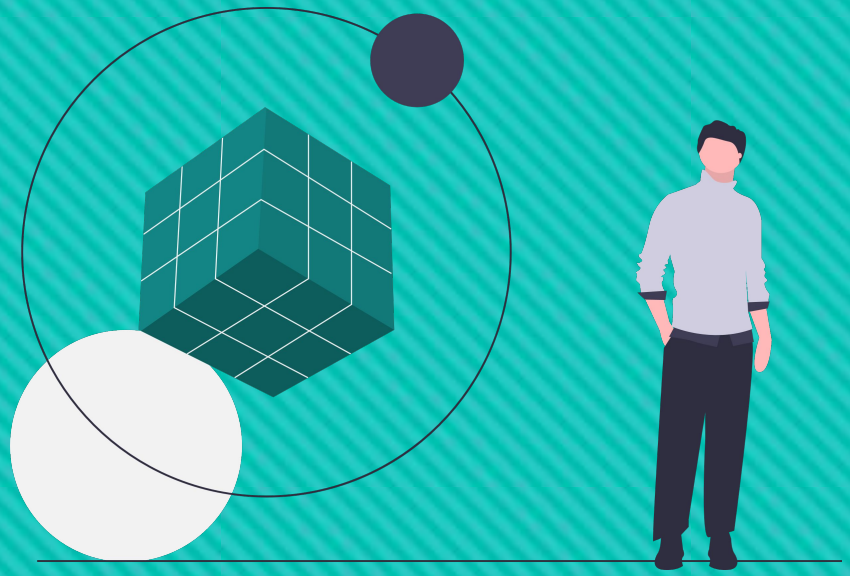






Conclusion

- ✔ From the above program and output, we conclude that the project is able to perform Visualization of Signal Transmission i.e. the Transmission of Electromagnetic Waves from an antennas or Satellites .
- ✔ Aim for the Project is Successfully Achieved !!!



Thank You