**Program in C – Divide your screen into four regions (axis) and draw circle,rectangle,ellipse and half ellipse in each region .**

#include<graphics.h>

#include<conio.h>

void main() {

int gd=DETECT,gm;

int midx,midy;

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

midx=getmaxx()/2;

midy=getmaxy()/2;

line(1,midy,getmaxx(),midy);

line(midx,1,midx,getmaxy());

circle(midx-200,midy-120,50);

rectangle(midx+100,midy-50,midx+200,midy-200);

ellipse(midx-180,midy+100,0,180,50,80);

ellipse(midx+180,midy+100,0,360,50,80);

getch();

closegraph();

}

**Program in C- to draw a simple hut**

#include<graphics.h>

#include<conio.h>

void main() {

int gd=DETECT,gm;

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

rectangle(150,180,250,300);

rectangle(250,180,420,300);

rectangle(180,250,220,300);

line(200,100,150,180);

line(200,100,250,180);

line(200,100,370,100);

line(370,100,420,180);

getch();

closegraph();

}

**C Program to generate Bresenham’s line drawaing algorithm**

#include<stdio.h>

#include<graphics.h>

#include<conio.h>

#include<math.h>

void main() {

int dx,dy,x,y,p,x1,x2,y1,y2;

int gd=DETECT,gm;

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

printf("Enter the x-coordinate of first point:x1:");

scanf("%d",&x1);

printf("Enter the y-coordinate of first point:y1:");

scanf("%d",&y1);

printf("Enter the x-coordinate of second point:x2:");

scanf("%d",&x2);

printf("Enter the y-coordinate of second point:y2:");

scanf("%d",&y2) ;

dx=abs(x2-x1);

dy=abs(y2-y1);

p=2\*(dy-dx);

x=x1;

y=y1;

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

putpixel(x,y,WHITE);

while(x<=x2) {

if(p<0)

{

x=x+1;

y=y;

p=p+2\*dy;

}

else

{

x=x+1;

y=y+1;

p=p+2\*(dy-dx);

}

putpixel(x,y,WHITE);

}

getch();

closegraph();

}

**C Program to generate DDA line drawaing algorithm**

#include<graphics.h>

#include<stdio.h>

#include<math.h>

#include<dos.h>

#include<conio.h>

void main() {

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

int i,gd=DETECT,gm ;

printf("Enter the value of x1:");

scanf("%f",&x1);

printf("Enter the value of y1:");

scanf("%f",&y1);

printf("Enter the value of x2:");

scanf("%f",&x2);

printf("Enter the value of y2:");

scanf("%f",&y2);

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

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,1);

x=x+dx;

y=y+dy;

i=i+1;

// sleep(1) ;

}

getch();

closegraph();

}

1. Write a C/C++ program to create a circle using midpoint circle algorithm

#include<graphics.h>

#include<stdio.h>

#include<conio.h>

void main() {

int xc,yc,r,pk,x,y;

int gd=DETECT,gm;

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

printf("Enter the x-coordinate of the center:xc:");

scanf("%d",&xc);

printf("Enter the y-coordinate of the center:yc:");

scanf("%d",&yc);

printf("Enter the radius:");

scanf("%d",&r);

x=0;

y=r;

pk=1-r;

while(x<y) {

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

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

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

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

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

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

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

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

if(pk<0)

{ x=x+1;

pk=pk+(2\*x)+3;

}

else

{

x=x+1;

y=y-1;

pk=pk+(2\*x)-(2\*y)+5;

}

}

getch();

closegraph();

}

1. Write a C/C++ program to perform 2D-translation .

#include<graphics.h>

#include<stdio.h>

#include<conio.h>

#include<dos.h>

void main() {

int gd=DETECT,gm;

int x1,y1,x2,y2,tx,ty;

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

printf("Enter endpoint x1:");

scanf("%d",&x1);

printf("Enter endpoint x2:");

scanf("%d",&x2);

printf("Enter endpoint y1:");

scanf("%d",&y1);

printf("Enter endpoint y2:");

scanf("%d",&y2);

line(x1,y1,x2,y2);

sleep(1);

printf("Enter translation coordinate tx:");

scanf("%d",&tx);

printf("Enter translation coordinate ty:");

scanf("%d",&ty);

x1=x1+tx;

y1=y1+ty;

x2=x2+tx;

y2=y2+ty;

printf("The new line after Translation:::");

line(x1,y1,x2,y2);

getch();

closegraph();

}

1. Write a C/C++ program to perform 2D-scaling

#include<graphics.h>

#include<stdio.h>

#include<conio.h>

#include<dos.h>

void main() {

int gd=DETECT,gm;

int x1,y1,x2,y2,sx,sy;

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

printf("Enter endpoint x1:");

scanf("%d",&x1);

printf("Enter endpoint x2:");

scanf("%d",&x2);

printf("Enter endpoint y1:");

scanf("%d",&y1);

printf("Enter endpoint y2:");

scanf("%d",&y2);

//setcolor(WHITE);

line(x1,y1,x2,y2);

sleep(1);

printf("Enter scaling coordinate sx:");

scanf("%d",&sx);

printf("Enter scaling coordinate sy:");

scanf("%d",&sy);

x1=x1\*sx;

y1=y1\*sy;

x2=x2\*sx;

y2=y2\*sy;

printf("The new line after scaling :::");

setcolor(WHITE);

line(x1,y1,x2,y2);

getch();

closegraph();

}

**C program for 2D reflection :**

#include<stdio.h>

#include<conio.h>

#include<graphics.h>

#include<dos.h>

void main() {

int gd=DETECT,gm;

int x1,y1,x2,y2,choice;

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

printf("Enter the endpoint x1 :");

scanf("%d",&x1);

printf("Enter the endpoint y1 :");

scanf("%d",&y1);

printf("Enter the endpoint x2 :");

scanf("%d",&x2);

printf("Enter the endpoint y2 :");

scanf("%d",&y2);

line(x1,y1,x2,y2);

sleep(1);

printf("1. Reflection about x-axis \n");

printf("2.Reflection about y-axis \n");

printf("Enter the choice: 1 - 2");

scanf("%d",&choice);

if(choice==1){

x1=x1;

x2=x2;

y1=y1+240;

y2=y2+240;

}

else if(choice==2) {

y1=y1;

y2=y2;

x1=x1+320;

x2=x2+320;

}

else

printf("Invalid choice");

line(x1,y1,x2,y2);

getch();

closegraph();

}

**C program for 2D shearing :**

#include<stdio.h>

#include<conio.h>

#include<graphics.h>

#include<dos.h>

void main() {

int gd=DETECT,gm;

int x1,y1,x2,y2,choice,shx,shy;

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

printf("Enter the endpoint x1 :");

scanf("%d",&x1);

printf("Enter the endpoint y1 :");

scanf("%d",&y1);

printf("Enter the endpoint x2 :");

scanf("%d",&x2);

printf("Enter the endpoint y2 :");

scanf("%d",&y2);

rectangle(x1,y1,x2,y2);

sleep(1);

printf("1. Shearing about x-axis \n");

printf("2. Shearing about y-axis \n");

printf("Enter the choice: 1 - 2");

scanf("%d",&choice);

if(choice==1){

printf("Enter the shearing factor: shx ");

scanf("%d",&shx);

x1=x1+(shx\*y1);

x2=x2+(shx\*y2);

y1=y1;

y2=y2;

}

else if(choice==2) {

printf("Enter the shearing factory: shy ");

scanf("%d",&shy);

y1=y1+(shy\*x1);

y2=y2+(shy\*x2);

x1=x1;

x2=x2;

}

else

printf("Invalid choice");

rectangle(x1,y1,x2,y2);

getch();

closegraph();

}

**Practical 8a:CohenSutherland Line Clipping in C**

**Algorithm**

**1.** Read 2 end points of line as p1(x1,y1) and p2(x2,y2)

**2.** Read 2 corner points of the clipping window (left-top and right-bottom) as (wx1,wy1) and (wx2,wy2)

**3.** Assign the region codes for 2 endpoints p1 and p2 using following steps:-

initialize code with 0000

Set bit 1 if x<wx1

Set bit 2 if x>wx2

Set bit 3 if y<wy2

Set bit 4 if y>wy1

**4.** Check for visibility of line

1. If region codes for both endpoints are zero then line is completely visible. Draw the line go to step 9.
2. If region codes for endpoints are not zero and logical ANDing of them is also nonzero then line is invisible. Discard the line and move to step 9.
3. If it does not satisfy 4.a and 4.b then line is partially visible.

**5.** Determine the intersecting edge of clipping window as follows:-

1. If region codes for both endpoints are nonzero find intersection points p1’ and p2’ with boundary edges.
2. If region codes for any one end point is non zero then find intersection point p1’ or p2’.

**6.** Divide the line segments considering intersection points.

**7.** Reject line segment if any end point of line appears outside of any boundary.

**8.** Draw the clipped line segment.

**9.** Stop.

#include<stdio.h>

#include<conio.h>

#include<math.h>

#include<graphics.h>

#include<dos.h>

struct point

{

int x,y;

char code[4];

};

void drawwindow();

void drawline(point p1,point p2);

point setcode(point p);

int visibility(point p1,point p2);

point resetendpt(point p1,point p2);

void main()

{

int gd=DETECT,v,gm;

point p1,p2,p3,p4,ptemp;

printf("\nEnter x1 and y1\n");

scanf("%d %d",&p1.x,&p1.y);

printf("\nEnter x2 and y2\n");

scanf("%d %d",&p2.x,&p2.y);

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

drawwindow();

delay(500);

drawline(p1,p2);

// delay(500);

getch();

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(500);

getch();

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(point p1,point p2)

{

line(p1.x,p1.y,p2.x,p2.y);

}

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

{

point 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(point p1,point 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);

}

point resetendpt(point p1,point p2)

{

point 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);

}

**Practical 8b: Liang Barsky Line Clipping Algorithm in C**

### Algorithm

**1.** Read 2 endpoints of line as p1 (x1, y1) & p2 (x2, y2).

**2.** Read 2 corners (left-top & right-bottom) of the clipping window as (xwmin, ywmin, xwmax, ywmax).

**3.** Calculate values of parameters pi and qi for i = 1, 2, 3, 4 such that

p1 = -dx, q1 = x1 – xwmin

p2 = dx, q2 = xwmax – x1

p3 = -dy, q3 = y1 – ywmin

p4 = dy, q4 = ywmax – y1

**4.** if pi = 0 then line is parallel to ith boundary

if qi < 0 then line is completely outside boundary so discard line

else, check whether line is horizontal or vertical and then check the line endpoints with the corresponding boundaries.

**5.** Initialize t1 & t2 as

t1 = 0 & t2 = 1

**6.** Calculate values for qi/pi for i = 1, 2, 3, 4.

**7.** Select values of qi/pi where pi < 0 and assign maximum out of them as t1.

**8.** Select values of qi/pi where pi > 0 and assign minimum out of them as t2.

**9.** if (t1 < t2)  
{  
xx1 = x1 + t1dx

xx2 = x1 + t2dx

yy1 = y1 + t1dy

yy2 = y1 + t2dy

line (xx1, yy1, xx2, yy2)  
}

**10.** Stop.

**PROGRAM :**

#include<stdio.h>

#include<graphics.h>

#include<math.h>

#include<dos.h>

void main()

{

int i,gd=DETECT,gm;

int x1,y1,x2,y2,xmin,xmax,ymin,ymax,xx1,xx2,yy1,yy2,dx,dy;

float t1,t2,p[4],q[4],temp;

x1=120;

y1=120;

x2=300;

y2=300;

xmin=100;

ymin=100;

xmax=250;

ymax=250;

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

rectangle(xmin,ymin,xmax,ymax);

dx=x2-x1;

dy=y2-y1;

p[0]=-dx;

p[1]=dx;

p[2]=-dy;

p[3]=dy;

q[0]=x1-xmin;

q[1]=xmax-x1;

q[2]=y1-ymin;

q[3]=ymax-y1;

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

{

if(p[i]==0)

{

printf("line is parallel to one of the clipping boundary");

if(q[i]>=0)

{

if(i<2)

{

if(y1<ymin)

{

y1=ymin;

}

if(y2>ymax)

{

y2=ymax;

}

line(x1,y1,x2,y2);

}

if(i>1)

{

if(x1<xmin)

{

x1=xmin;

}

if(x2>xmax)

{

x2=xmax;

}

line(x1,y1,x2,y2);

}

}

}

}

t1=0;

t2=1;

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

{

temp=q[i]/p[i];

if(p[i]<0)

{

if(t1<=temp)

t1=temp;

}

else

{

if(t2>temp)

t2=temp;

}

}

if(t1<t2)

{

xx1 = x1 + t1 \* p[1];

xx2 = x1 + t2 \* p[1];

yy1 = y1 + t1 \* p[3];

yy2 = y1 + t2 \* p[3];

line(xx1,yy1,xx2,yy2);

}

getch();

closegraph();

}

**Pract 9a: Write a program to fill a circle using boundary fill algorithm**

This algorithm picks a point inside an figure and starts to fill until it reaches the boundary of the figure. The color of the boundary and the color that we fill should be different for this algorithm to [work.In](http://work.in/" \t "_blank) this algorithm, we assume that color of the boundary is same for the entire figure. The boundary fill algorithm can be implemented by 4-connected pixels or 8-connected pixels.

**Program :**

#include<stdio.h>

#include<graphics.h>

#include<conio.h>

#include<dos.h>

void boundaryfill(int x,int y,int fillcolor,int bordercolor) {

if(getpixel(x,y)!=bordercolor&&getpixel(x,y)!=fillcolor) {

putpixel(x,y,fillcolor);

boundaryfill(x+1,y,fillcolor,bordercolor);

boundaryfill(x,y+1,fillcolor,bordercolor);

boundaryfill(x-1,y,fillcolor,bordercolor);

boundaryfill(x,y-1,fillcolor,bordercolor);

}

}

void main() {

int gd=DETECT,gm,radius ;

int x,y;

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

printf("Enter the center of the circle:\n") ;

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

printf("Enter the radius:");

scanf("%d",&radius);

circle(x,y,radius);

boundaryfill(x,y,5,15);

getch();

closegraph();

}

**Pract 9b: Write a program to fill a circle using flood fill algorithm**

1. The limitations of boundary fill algorithm like if the polygon is having boundaries with different colors then boundary fill algorithm fails.
2. This limitation of boundary fill algorithm is overcome in flood fill algorithm.
3. The flood fill algorithm is also called as seed fill algorithm.
4. This algorithm begins with seed point.
5. Instead of checking boundary color this algorithm checks whether the pixel is having the polygon’s original color i.e. previous or old color.
6. If the answer is yes then fills that pixel with new color and use each of the pixels neighboring pixel as a new seed in a recursive call.
7. If the answer is no i.e. the color of pixel is already changed then return to its caller.
8. Sometimes we want to fill an area that is not defined within a single color boundary .
9. We can paint such areas by replacing a specified interior color instead of searching for a boundary color value.
10. Here we are setting empty pixel with new color till we get any colored pixel.
11. Flood fill and boundary fill algorithms are somewhat similar.
12. A flood fill algorithm is particular used when the region or polygon has no uniformed colored boundaries.

**Disadvantages of Seed Fill Algorithm:**

1) If an inside pixel is in some other color then the fill terminates and the polygon remains unfilled.

2) Seed fill method does not work for large polygons.

**Program**

#include<graphics.h>

#include<stdio.h>

#include<conio.h>

#include<dos.h>

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

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

{

delay(10);

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);

}

}

void main() {

int gd=DETECT,gm,r ;

int x,y;

printf("Enter the x and y cordinate of the center of the circle

:");

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

printf("Enter the radius of the circle :");

scanf("%d",&r);

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

circle(x,y,r);

floodFill(x,y,0,9);

getch();

closegraph();

}

settextxy(int font,int direction,int fontsize)

1. Font and different options are as follow:

DEFAULT\_FONT 0

TRIPLEX\_FONT 1

SMALL\_FONT 2

SANS\_SERIF\_FONT 3

GOTHIC\_FONT 4

SCRIPT\_FONT 5

SIMPLEX\_FONT 6

TRIPLEX\_SCR\_FONT 7

COMPLEX\_FONT 8

EUROPEAN\_FONT 9

BOLD\_FONT 10

1. Font direction is either 0: left to right(HORIZ\_DIR) or 1:bottom to top(VERT\_DIR)
2. Font size: it controls the size of the font . values 1-10 ;

**Practical 10 a: Develop a simple text screen saver using simple graphics function**

#include<stdio.h>

#include<graphics.h>

#include<conio.h>

void main() {

int gd=DETECT,gm,x,i ;

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

for(x=0;x<500;x++)

{

cleardevice();

settextstyle(1,0,5);

setcolor(RED);

outtextxy(50,415-x,"Welcome");

setcolor(GREEN);

outtextxy(250,415-x,"to");

setcolor(YELLOW);

settextstyle(3,0,5);

outtextxy(350,415-x,"Graphics");

}

getch();

closegraph();

}

**10b .Perform smiling face animation using graphics function**

#include<graphics.h>

#include<stdio.h>

#include<conio.h>

#include<dos.h>

void main() {

int gd=DETECT,gm,i ;

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

for(i=1;i<=10;i++) {

cleardevice();

circle(200,200,30); //head

circle(190,190,5); //for left eye

arc(190,192,50,130,10);

circle(210,190,5); //for right eye

arc(210,192,50,130,10);

//arc(190,192,50,130,10);

//for smiling lips

if(i%2==0) {

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

line(187,210,193,212);

line(207,210,213,212);

}

//for not smiling

else {

line(193,205,193,215);

line(193,210,207,210);

line(207,205,207,215);

}

delay(500);

}

getch();

closegraph();

}

**10c. Draw the moving car on the screen**

#include<stdio.h>

#include<graphics.h>

#include<conio.h>

#include<dos.h>

void main() {

int gd=DETECT,gm ;

int i,maxx,midy;

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

maxx=getmaxx();

midy=getmaxy()/2;

for(i=0;i<maxx-150;i=i+5) {

cleardevice();

setcolor(WHITE);

line(1,midy+37,maxx,midy+37);

setcolor(YELLOW);

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);

circle(30+i,midy+25,9);

circle(90+i,midy+25,9);

delay(100);

}

getch();

closegraph();

}

**11. write a c++ program to draw concetric circle & fill it with different color**

#include<stdio.h>

#include<conio.h>

#include<graphics.h>

int main() {

int gd=DETECT,gm;

int x,y,r=100,r1=80,r2=60,r3=40;

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

x=getmaxx()/2; y=getmaxy()/2;

setcolor(4) ;

cirlce(x,y,r);

setcolor(8) ;

cirlce(x,y,r1);

setcolor(3) ;

cirlce(x,y,r2);

setcolor(7) ;

cirlce(x,y,r3);

getch();

closegraph();

return 0;

}

**12.Write a C progrom for Bouncing ball Animation**

#include<stdio.h>

#include<conio.h>

#include<graphics.h>

#include<dos.h>

void main()

{

int gd=DETECT,gm=DETECT;

int x,y=0,j,t=400,c=1;

initgraph(&gd,&gm,"c:\\turboC3\\bgi");

setcolor(RED);

setfillstyle(SOLID\_FILL,RED);

for(x=40;x<602;x++)

{

cleardevice();

circle(x,y,30);

floodfill(x,y,RED);

delay(40);

if(y>=400)

{

c=0;

t-=20;

}

if(y<=(400-t))

c=1;

y=y+(c?15:-15);

}

getch();

}

**C Program to display concentric circles :**

#include<stdio.h>

#include<conio.h>

#include<graphics.h>

int main() {

int gd=DETECT,gm;

int x,y,r=100,r1=80,r2=60,r3=40;

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

x=getmaxx()/2; y=getmaxy()/2;

setcolor(4) ;

circle(x,y,r);

setcolor(8) ;

circle(x,y,r1);

setcolor(3) ;

circle(x,y,r2);

setcolor(7) ;

circle(x,y,r3);

getch();

closegraph();

return 0;

}