**1.DDA Line**

#include <iostream>

#include <cmath>

#include <graphics.h> // Include graphics library

void ddaLine(int x0, int y0, int x1, int y1) {

    int dx = x1 - x0;

    int dy = y1 - y0;

    int steps = std::max(abs(dx), abs(dy));

    float xIncrement = dx / static\_cast<float>(steps);

    float yIncrement = dy / static\_cast<float>(steps);

    float x = x0;

    float y = y0;

    for (int i = 0; i <= steps; ++i) {

        putpixel(round(x), round(y), WHITE); // Plot the pixel

        x += xIncrement;

        y += yIncrement;

    }

}

int main() {

    int gd = DETECT, gm;

    initgraph(&gd, &gm, ""); // Initialize graphics mode

    int x0 = 100, y0 = 100, x1 = 200, y1 = 200;

    ddaLine(x0, y0, x1, y1);

    getch(); // Wait for a key press

    closegraph(); // Close graphics mode

    return 0;

}

**2.Bresenham Circle**

   #include<graphics.h>

#include<iostream>

#include<math.h>

using namespace std;

void bresenham\_circle(int r, int xc , int yc)

{

int x=0,y=r;

int d = 3-2\*r;

while(x<=y)

{

     if (d>=0)

     {d+= 4\*(x-y)+10;

      x++;

      y--;

     }

     else if(d<0)

      {d+= 4\*(x) + 6;

      x++;

     }

     putpixel(xc+x,y+yc,WHITE);

     putpixel(-x+xc,y+yc,WHITE);

     putpixel(x+xc,-y+yc,WHITE);

     putpixel(-x+xc,-y+yc,WHITE);

     putpixel(y+xc,x+yc,WHITE);

     putpixel(-y+xc,x+yc,WHITE);

     putpixel(y+xc,-x+yc,WHITE);

     putpixel(-y+xc,-x+yc,WHITE);

     delay(100);

}

}

int main()

{

    int gd = DETECT, gm;

    int r, xc, yc;

    cout<< "Enter the r,xc,yc :"<<endl;

    cin>>r>>xc>>yc;

    initgraph(&gd,&gm,NULL);

    bresenham\_circle(r,xc,yc);

    getch();

    closegraph();

    return 0;

}

**3.Midpoint circle**

#include<graphics.h>

#include<iostream>

#include<math.h>

using namespace std;

void mid\_circle(int r, int xc , int yc)

{

int x=0,y=r;

float d = (5/4)-r;

while(x<=y)

{

     if (d>=0)

     {d+= 2\*(x-y+5);

      x++;

      y--;

     }

     else if(d<0)

      {d+= 2\*(x) + 3;

      x++;

     }

     putpixel(xc+x,y+yc,WHITE);

     putpixel(-x+xc,y+yc,WHITE);

     putpixel(x+xc,-y+yc,WHITE);

     putpixel(-x+xc,-y+yc,WHITE);

     putpixel(y+xc,x+yc,WHITE);

     putpixel(-y+xc,x+yc,WHITE);

     putpixel(y+xc,-x+yc,WHITE);

     putpixel(-y+xc,-x+yc,WHITE);

     delay(100);

}

}

int main()

{

    int gd = DETECT, gm;

    int r, xc, yc;

    cout<< "Enter the r,xc,yc :"<<endl;

    cin>>r>>xc>>yc;

    initgraph(&gd,&gm,NULL);

    mid\_circle(r,xc,yc);

    getch();

    closegraph();

    return 0;

}

**4.Boundaryfill**

#include <graphics.h>

#include <iostream>

using namespace std;

void boundaryfill ( int x, int y, int b, int n)

{

if (get pixel (x, y) != b && get pixel (x,y)!=n)

{

putpixel (x, y, n);

boundaryfill (x+1, y,b,n);

boundaryfill (x-1, y, b, n);

boundaryfill (x, y+l, b,n);

boundaryfill (x, y-1, b, n);

 }

}

int main ()

{

int gd ,DETECT, gm;

initgraph(&gd, &gm, NULL);

line (100, 100, 50,400);

line (100, 100, 150, 400);

line (50, 400, 100, 2001);

line (150, 400,100,200);

boundaryfill (100, 150, 7, 14);

getch();

closegraph();

return 0;

}

**5.Floodfill**

#include < graphics.h>

#include <iostream>

using namespace std;

void floodfill (int x, int y, int a, int n)

{

if (get pixel (x, y) ==0)

{

put pixel (x, y, n);

floodfill (x + 1, y, 0, n);

floodfill (x-1, y,0, n);

floodfill (x, y+1, o, n);

floodfill (x, y-l, o, n);

floodfill (x+1, y-1, 0, n);

floodfill (x + 1, y + 1, 0, n)

floodfill (x- 1, y + 1, 0, n) ;

floodfill (x - 1, y - 1, 0, n)

}

}

int main ()

{

int gd = DETECT, gm;

initgraph (&gd, &gm, NULL);

line (100, 100, 50,400);

line (100, 100, 150, 400);

line (50, 400, 100, 200);

line (150, 400, 100, 200);

floodfill (100, 150, 0,7);

getch();

close graph();

return 0;

}

**6.**