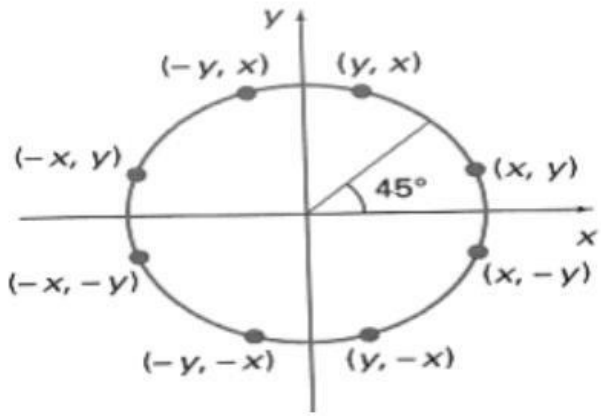
Aim: To implement midpoint circle algorithm.

Objective:

Draw a circle using mid-point circle drawing algorithm by determining the points needed for rasterizing a circle. The mid-point algorithm to calculate all the perimeter points of the circle in the first octant and then print them along with their mirror points in the other octants.

Theory:

The shape of the circle is similar in each quadrant. We can generate the points in one section and the points in other sections can be obtained by considering the symmetry about x-axis and y-axis.



The equation of circle with center at origin is x 2 + y 2 = r 2

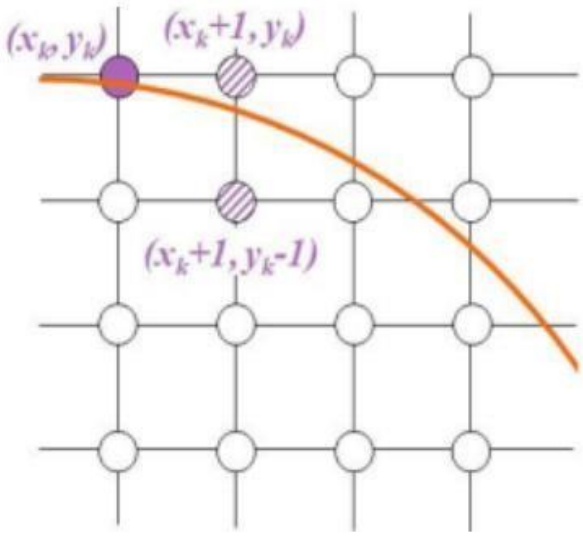
Let the circle function is f circle (x, y) - is <

0, if (x, y) is inside circle boundary, is = 0, if

(x, y) is on circle boundary, is > 0, if (x, y)

is outside circle boundary.

Consider the pixel at (xk, yk) is plotted,



Now the next pixel along the circumference of the circle will be either (xk + 1, yk) or (xk + 1, yk – 1) whichever is closer the circle boundary.

Let the decision parameter pk is equal to the circle function evaluate at the mid-point between two pixels.

If pk < 0, the midpoint is inside the circle and the pixel at yk is closer to the circle boundary. Otherwise, the midpoint is outside or on the circle boundary and the pixel at yk – 1 is closer to the circle boundary.

Algorithm –

MIDPOINT_CIRCLE(r)

x=0 y=r p=1-r

EightWaySummetry(x,y)

putpixel(x,y) putpixel(x,-y)

putpixel(-x,y) putpixel(-x,-y)

MIDPOINT_CIRCLE(r) x=0 y=r

p=1-r

EightWaySummetry(x,y)

putpixel(x,y) putpixel(x,-y)

putpixel(-x,y) putpixel(-x,-

y) putpixel(y,x) putpixel(y,-

x) putpixel(-y,x) putpixel(-

y,-x) while x<y do if p<0

p=p+2x+3

else p=p+2x-

2y+5 y=y-1 end

x=x+1 end

Program –

```
#include<stdio.h>

#include<conio.h>

#include<graphics.h> void
main()

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

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

  printf("Enter xc,yc:");

scanf("%d%d",&xc,&yc);

printf("Enter radius:");

scanf("%d",&r);

  x=0;   y=r;

p=1-r;   do
```

```
 {
    putpixel(xc+x,yc+y,RED);
putpixel(xc+y,yc+x,RED);
putpixel(xc-y,yc+x,RED);
putpixel(xc-x,yc+y,RED);
putpixel(xc-x,yc-y,RED);
putpixel(xc-y,yc-x,RED);
putpixel(xc+y,yc-x,RED);
putpixel(xc+x,yc-y,RED);
if(p<0)
    {
    p=p+(2*x)+3;
    }     else
{

    y=y-1;      p=p+(2*x)-
(2*y)+1;
    }
    x=x+1;      }while(y>x);
getch();      closegraph();
}
```

Output:

Conclusion: Comment on

1. Fast or slow:-It is slow
2. Draw one arc only and repeat the process in 8 quadrants:-It is quite easy to draw only one arc rather than drawing all arcs
3. Difference with line drawing method:-It is difficult than line drawing algorithms