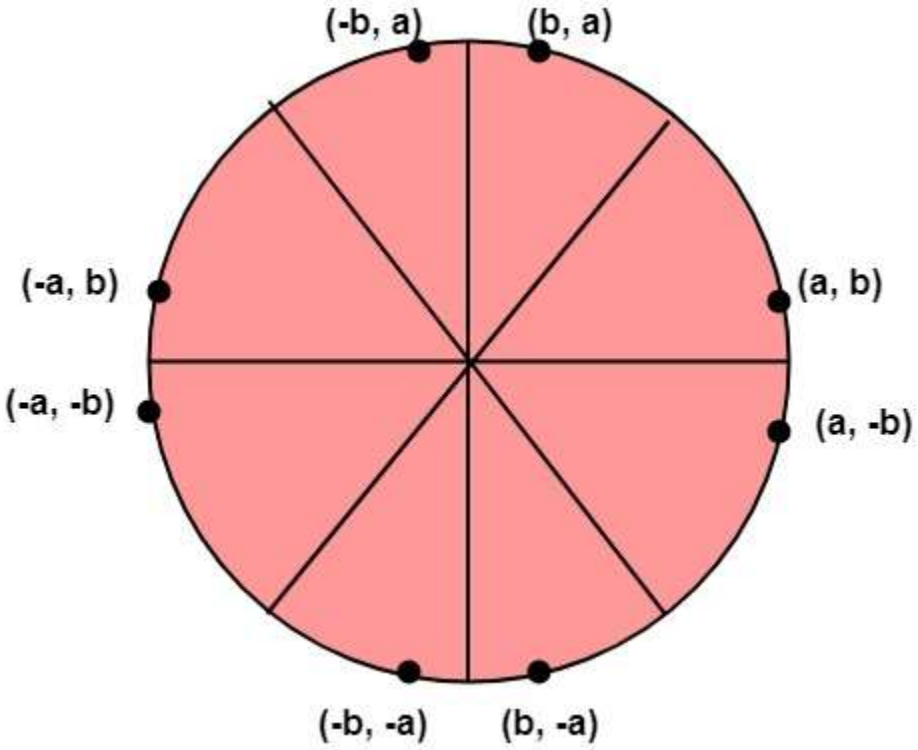
MidPoint Circle Algorithm

It is based on the following function for testing the spatial relationship between the arbitrary point (x, y) and a circle of radius r centered at the origin:

$$f(x, y) = x^2 + y^2 - r^2 \quad \begin{bmatrix} < 0 \text{ for } (x,y) \text{inside the circle} \\ = 0 \text{ for } (x,y) \text{on the circle} \\ > 0 \text{ for } (x,y) \text{outside the circle} \end{bmatrix} \text{.....equation 1}$$



Now, consider the coordinates of the point halfway between pixel T and pixel S

This is called midpoint $(x_{i+1}, y_i - \frac{1}{2})$ and we use it to define a decision parameter:

$$P_i = f(x_{i+1}, y_i - \frac{1}{2}) = (x_{i+1})^2 + (y_i - \frac{1}{2})^2 - r^2 \text{ .....equation 2}$$

If P_i is -ve $\Rightarrow$midpoint is inside the circle and we choose pixel T

If P_i is+ve $\Rightarrow$midpoint is outside the circle (or on the circle)and we choose pixel S.

The decision parameter for the next step is:

$$P_{i+1} = (x_{i+1} + 1)^2 + (y_{i+1} - \frac{1}{2})^2 - r^2 \text{.....equation 3}$$

Since $x_{i+1}=x_{i+1}$, we have

$$P_{i+1} - P_i = ((x_i + 1) + 1)^2 - (x_i + 1)^2 + (y_{i+1} - \frac{1}{2})^2 - (y_i - \frac{1}{2})^2$$

$$= x_i^2 + 4 + 4x_i - x_i^2 + 1 - 2x_i + y_{i+1}^2 + \frac{1}{4} - y_{i+1} - y_i^2 - \frac{1}{4} - y_i$$

$$= 2(xi+1) + 1 + (y_{i+1}^2 - y_i^2) - (y_{i+1} - y_i)$$

$$P_{i+1} = P_i + 2(x_i + 1) + 1 + (y_{i+1}^2 - y_i^2) - (y_{i+1} - y_i) \quad \text{.....equation 4}$$

If pixel T is choosen $\Rightarrow P_i<0$

We have $y_{i+1}=y_i$

If pixel S is choosen $\Rightarrow P_i \geq 0$

We have $y_{i+1}=y_i-1$

$$\text{Thus,} \quad P_{i+1} = \begin{bmatrix} P_i + 2(x_i + 1) + 1, & \text{if } P_i < 0 \\ P_i + 2(x_i + 1) + 1 - 2(y_i - 1), & \text{if } P_i \geq 0 \end{bmatrix} \text{.....equation 5}$$

We can continue to simplify this in n terms of (x_i,y_i) and get

$$P_{i+1} = \begin{bmatrix} P_i + 2x_i + 3, & \text{if } P_i < 0 \\ P_i + 2(x_i - y_i) + 5, & \text{if } P_i \geq 0 \end{bmatrix} \text{.....equation 6}$$

Now, initial value of P_i (0,r)from equation 2

$$P_1 = (0 + 1)^2 + (r - \frac{1}{2})^2 - r^2$$

$$= 1 + \frac{1}{4} - r^2 = \frac{5}{4} - r$$

We can put $\frac{5}{4} \cong 1$
$\therefore r$ is an integer
So, $P_1=1-r$

Algorithm:

Step1: Put x =0, y =r in equation 2
We have p=1-r

Step2: Repeat steps while x ≤ y
Plot (x, y)

If (p<0)
Then set p = p + 2x + 3
Else
p = p + 2(x-y)+5
y =y - 1 (end if)
x =x+1 (end loop)

Step3: End

Program to draw a circle using Midpoint Algorithm:

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

class bresen
{
   float x, y,a, b, r, p;
   public:
   void get ();
   void cal ();
};
   void main ()
   {
   bresen b;
   b.get ();
   b.cal ();
   getch ();
  }
   Void bresen :: get ()
  {
   cout<<"ENTER CENTER AND RADIUS";
    cout<< "ENTER (a, b)";
   cin>>a>>b;
   cout<<"ENTER r";
   cin>>r;
}
void bresen ::cal ()
{
   /* request auto detection */
   int gdriver = DETECT,gmode, errorcode;
   int midx, midy, i;
   /* initialize graphics and local variables */
   initgraph (&gdriver, &gmode, " ");
   /* read result of initialization */
   errorcode = graphresult ();
   if (errorcode ! = grOK)    /*an error occurred */
   {
```

```
41.         printf("Graphics error: %s \n", grapherrormsg (errorcode);
42.         printf ("Press any key to halt:");
43.         getch ();
44.         exit (1); /* terminate with an error code */
45.      }
46.      x=0;
47.      y=r;
48.      putpixel (a, b+r, RED);
49.      putpixel (a, b-r, RED);
50.      putpixel (a-r, b, RED);
51.      putpixel (a+r, b, RED);
52.      p=5/4)-r;
53.      while (x<=y)
54.      {
55.         If (p<0)
56.         p+= (4*x)+6;
57.         else
58.         {
59.            p+=(2*(x-y))+5;
60.            y--;
61.         }
62.         x++;
63.         putpixel (a+x, b+y, RED);
64.         putpixel (a-x, b+y, RED);
65.         putpixel (a+x, b-y, RED);
66.         putpixel (a+x, b-y, RED);
67.         putpixel (a+x, b+y, RED);
68.         putpixel (a+x, b-y, RED);
69.         putpixel (a-x, b+y, RED);
70.         putpixel (a-x, b-y, RED);
71.      }
72.   }
```

Output:

ENTER CENTER AND RADIUS

ENTER (a, b) 319, 239

ENTER r 100