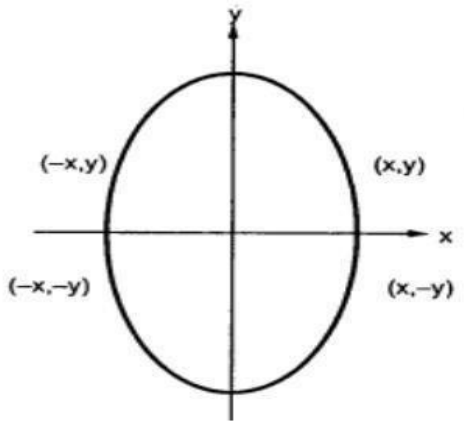
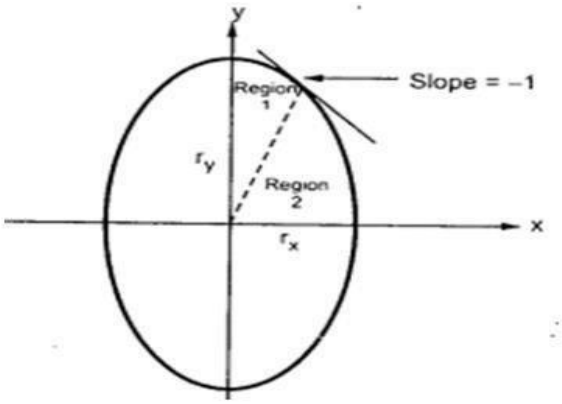
Aim-To implement midpoint Ellipse algorithm **Objective:**

Draw the ellipse using Mid-point Ellipse algorithm in computer graphics. Midpoint ellipse algorithm plots (finds) points of an ellipse on the first quadrant by dividing the quadrant into two regions.

Theory:

Midpoint ellipse algorithm uses four way symmetry of the ellipse to generate it. Figure shows the 4-way symmetry of the ellipse.





Here the quadrant of the ellipse is divided into two regions as shown in the fig. Fig. shows the division of first quadrant according to the slope of an ellipse with rx < ry. As ellipse is drawn from 90 0 to 0 0 , x moves in positive direction and y moves in negative direction and ellipse passes through two regions 1 and 2.

The equation of ellipse with center at (xc, yc) is given as -

[(x – xc) / rx] 2 + [(y – yc) / ry] 2 = 1

Therefore, the equation of ellipse with center at origin is given as -

[x / rx] 2 + [y / ry] 2 = 1

i.e. x 2 ry 2 + y 2 rx 2 = rx 2 ry 2

Let, f ellipse (x, y) = x2 ry2 + y2 rx2 - rx2 ry2

Algorithm:

1. Initialize the ellipse parameters:

- Center (xc, yc)

- Semimajor axis length 'a'

- Semiminor axis length 'b'

2. Initialize initial decision parameters:

- Decision parameter for region 1: p1 = b^2 - a^2*b + 0.25*a^2

- Decision parameter for region 2: p2 = b^2 * (x + 0.5)^2 + a^2 * (y - 1)^2 - a^2 * b^2

3. Set initial point (x, y) in the first octant:

- x = 0

- y = b

4. Iterate while a^2 * (y - 0.5) > b^2 * (x + 1):

- If p1 < 0, increment x and update p1: p1 = p1 + b^2 * (2 * x + 3)

- If p1 >= 0, increment x and decrement y, update p1: p1 = p1 + b^2 * (2 * x + 3) + a^2 * (1 - 2 * y)

5. Update p2 for region 2: p2 = p2 - a^2 * (2 * y - 3)

6. Plot the obtained points in the other octants using symmetry:

- Plot points (xc + x, yc + y), (xc - x, yc + y), (xc + x, yc - y), and (xc - x, yc - y)

7. Repeat steps 4-6 until x >= a.

8. After finishing the first quadrant, use symmetry to plot points in the other quadrants using the calculated points.

9. The ellipse is drawn using the calculated points in all octants.

10. Terminate.

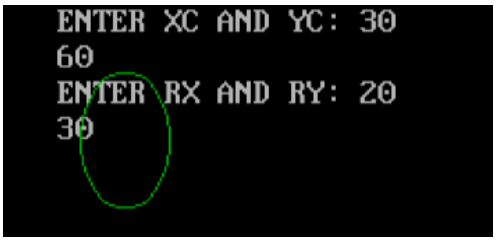
Program:

```
#inlcude <stdio.h>
#include<graphics.h>
#include<conio.h>
#include<math.h>
void main()
{

int p,rx,ry,xc,yc,x,y;
int gd= DETECT, gm; initgraph(&gd,
&gm, "C:\\TURBOC3\\BGI");
printf("Enter xc, yc"); scanf("%d
%d", &xc, &yc); printf("Enter rx,
ry"); scanf("%d %d", &rx, &ry); x=0;
y = ry; p = (ry*ry)- (ry * rx * rx)+((rx *
rx)/4);
while((2*x*ry*ry)<(2*y*rx*rx))
{
putpixel(xc + x, yc +y, RED);
putpixel(xc - x, yc +y, RED);
putpixel(xc + x, yc -y, RED);
putpixel(xc - x, yc -y, RED); if(p<0)
{
p = p+(2*x*ry*ry) + (ry *
ry); x = x + 1;
}
```

```
else{
p = p+ (2*x*ry*ry) + (ry * ry) - (2*y*rx*rx);  x=
x+1;
y= y - 1;
}
}
p = (x +0.5)* (x +0.5)*(ry * ry) *(y-1)*(y-1)-(rx * rx * ry * ry);  while(y>=0)
{
putpixel(xc + x, yc +y, RED);
putpixel(xc - x, yc +y, RED);
putpixel(xc + x, yc -y, RED);
putpixel(xc - x, yc -y, RED); if(p<0)
{  p = p+ (rx * rx) - (2 * rx *
rx * y); y= y-1;
}
else
{  p = p + (rx * rx) - (2 * rx * rx * y) + ( 2* ry *
ry * x); y= y-1; x = x+1;
}
getch();
closegraph();
}
```

Output:



Conclusion:

1. **Slow or Fast**: The midpoint ellipse algorithm is generally considered fast. Its time complexity is proportional to the number of pixels needed to draw the

ellipse, which is much faster than other methods that involve complex mathematical calculations. However, its speed may vary depending on the hardware and software implementation. When optimized for integer arithmetic and hardware acceleration, it can be exceptionally fast for drawing ellipses on computer screens.

2. **Difference with Circle**: The midpoint ellipse algorithm differs from drawing a circle primarily in how it calculates and plots points. While both shapes involve plotting points symmetrically around a center, the key difference lies in the decision parameter and how it's updated. In the midpoint circle algorithm, a fixed decision parameter is used, resulting in perfect circular shapes. In contrast, the midpoint ellipse algorithm adjusts the decision parameter to account for the ellipse's eccentricity, which allows it to accurately draw ellipses with varying aspect ratios.
3. **Importance of Object**: The importance of implementing the midpoint ellipse algorithm lies in its versatility and relevance in computer graphics, image processing, and various graphical applications. Ellipses are common geometric shapes, and accurately rendering them is crucial in fields such as graphic design, game development, CAD (Computer-Aided Design), and scientific visualization. The algorithm provides a straightforward and efficient way to draw ellipses, making it an essential tool for creating visually appealing and accurate graphical representations of objects and structures in these domain