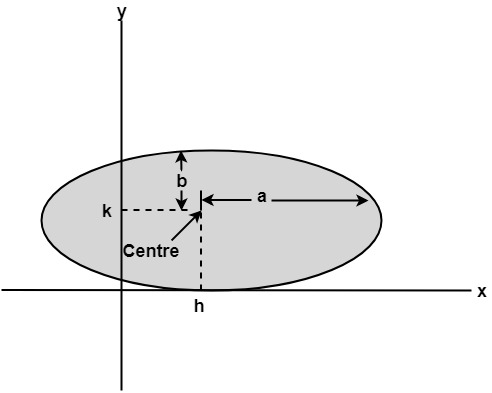
**Scan Converting a Ellipse:**

The ellipse is also a symmetric figure like a circle but is four-way symmetry rather than eight-way.



**Program to Implement Ellipse Drawing Algorithm:**

1. #include<stdio.h>
2. #include<conio.h>
3. #include<graphics.h>
4. #include<math.h>
5. void disp();
6. float x,y;
7. intxc,yc;
8. void main()
9. {
10. intgd=DETECT,gm,a,b;
11. float p1,p2;
12. clrscr();
13. initgraph(&gd,&gm,"c:\\turboc3\\bgi");
14. printf("\*\*\* Ellipse Generating Algorithm \*\*\*\n");
15. printf("Enter the value of Xc\t");
16. scanf("%d",&xc);
17. printf("Enter the value of yc\t");
18. scanf("%d",&yc);
19. printf("Enter X axis length\t");
20. scanf("%d",&a);
21. printf("Enter Y axis length\t");
22. scanf("%d",&b);
23. x=0;y=b;
24. disp();
25. p1=(b\*b)-(a\*a\*b)+(a\*a)/4;
26. while((2.0\*b\*b\*x)<=(2.0\*a\*a\*y))
27. {
28. x++;
29. if(p1<=0)
30. p1=p1+(2.0\*b\*b\*x)+(b\*b);
31. else
32. {
33. y--;
34. p1=p1+(2.0\*b\*b\*x)+(b\*b)-(2.0\*a\*a\*y);
35. }
36. disp();
37. x=-x;
38. disp();
39. x=-x;
40. delay(50);
41. }
42. x=a;
43. y=0;
44. disp();
45. p2=(a\*a)+2.0\*(b\*b\*a)+(b\*b)/4;
46. while((2.0\*b\*b\*x)>(2.0\*a\*a\*y))
47. {
48. y++;
49. if(p2>0)
50. p2=p2+(a\*a)-(2.0\*a\*a\*y);
51. else
52. {
53. x--;
54. p2=p2+(2.0\*b\*b\*x)-(2.0\*a\*a\*y)+(a\*a);
55. }
56. disp();
57. y=-y;
58. disp();
59. y=-y;
60. delay(50);
61. }
62. getch();
63. closegraph();
64. }
65. void disp()
66. {
67. putpixel(xc+x,yc+y,7);
68. putpixel(xc-x,yc+y,7);
69. putpixel(xc+x,yc-y,7);
70. putpixel(xc+x,yc-y,7);
71. }

**Output:**

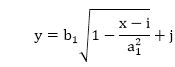


There two methods of defining an Ellipse:

1. Polynomial Method of defining an Ellipse
2. Trigonometric method of defining an Ellipse

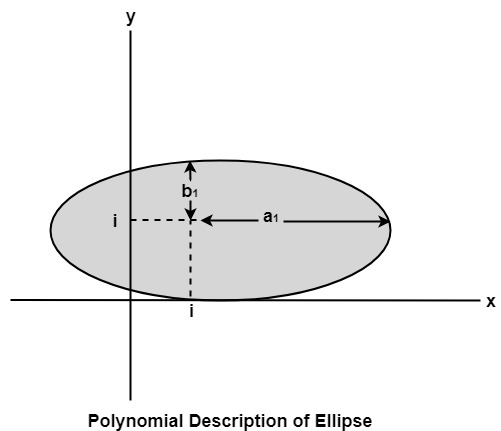
# Polynomial Method:

The ellipse has a major and minor axis. If a1 and b1are major and minor axis respectively. The centre of ellipse is (i, j). The value of x will be incremented from i to a1and value of y will be calculated using the following formula



## Drawback of Polynomial Method:

1. It requires squaring of values. So floating point calculation is required.
2. Routines developed for such calculations are very complex and slow.



## Algorithm:

1. Set the initial variables: a = length of major axis; b = length of minor axis; (h, k) = coordinates of ellipse center; x = 0; i = step; xend = a.

2. Test to determine whether the entire ellipse has been scan-converted. If x>xend, stop.

3. Compute the value of the y coordinate:

Polynomial Method

4. Plot the four points, found by symmetry, at the current (x, y) coordinates:

          Plot (x + h, y + k)           Plot (-x + h, -y + k)           Plot (-y - h, x + k)           Plot (y + h, -x + k)

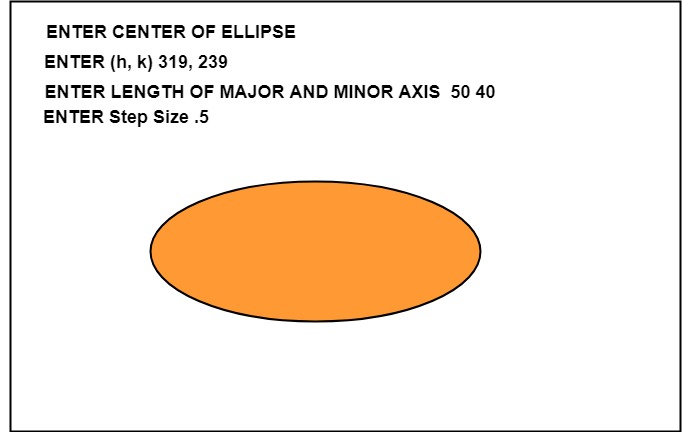
5. Increment x; x = x + i.

6. Go to step 2.

### Program to draw an Ellipse using Polynomial Method:

1. #include <graphics.h>
2. #include <stdlib.h>
3. #include <math.h>
4. #include <stdio.h>
5. #include <conio.h>
6. #include <iostream.h>
8. class bresen
9. {
10. float x, y, a, b, r, t, te, xend, h, k, step;
11. public:
12. void get ();
13. void cal ();
14. };
15. void main ()
16. {
17. bresen b;
18. b.get ();
19. b.cal ();
20. getch ();
21. }
22. void bresen :: get ()
23. {
24. cout<<"\n ENTER CENTER OF ELLIPSE";
25. cout<<"\n enter (h, k) ";
26. cin>>h>>k;
27. cout<<"\n ENTER LENGTH OF MAJOR AND MINOR AXIS";
28. cin>>a>>b;
29. cout<<"\n ENTER Step Size";
30. cin>> step;
31. }
32. void bresen ::cal ()
33. {
34. /\* request auto detection \*/
35. int gdriver = DETECT,gmode, errorcode;
36. int midx, midy, i;
37. /\* initialize graphics and local variables \*/
38. initgraph (&gdriver, &gmode, " ");
39. /\* read result of initialization \*/
40. errorcode = graphresult ();
41. if (errorcode ! = grOK)    /\*an error occurred \*/
42. {
43. printf("Graphics error: %s \n", grapherrormsg (errorcode);
44. printf ("Press any key to halt:");
45. getch ();
46. exit (1); /\* terminate with an error code \*/
47. }
48. x = 0;
49. xend=a;
50. whilex (x<xend)
51. {
52. t= (1-((x \* x)/ (a \* a)));
53. if (t<0)
54. te=-t;
55. else
56. te=t;
57. y=b \* sqrt (te);
58. putpixel (h+x, k+y, RED);
59. putpixel (h-x, k+y, RED);
60. putpixel (h+x, y-y, RED);
61. putpixel (h-x, k-y, RED);
62. x+=step;
63. }
64. getch();
65. }

**Output:**

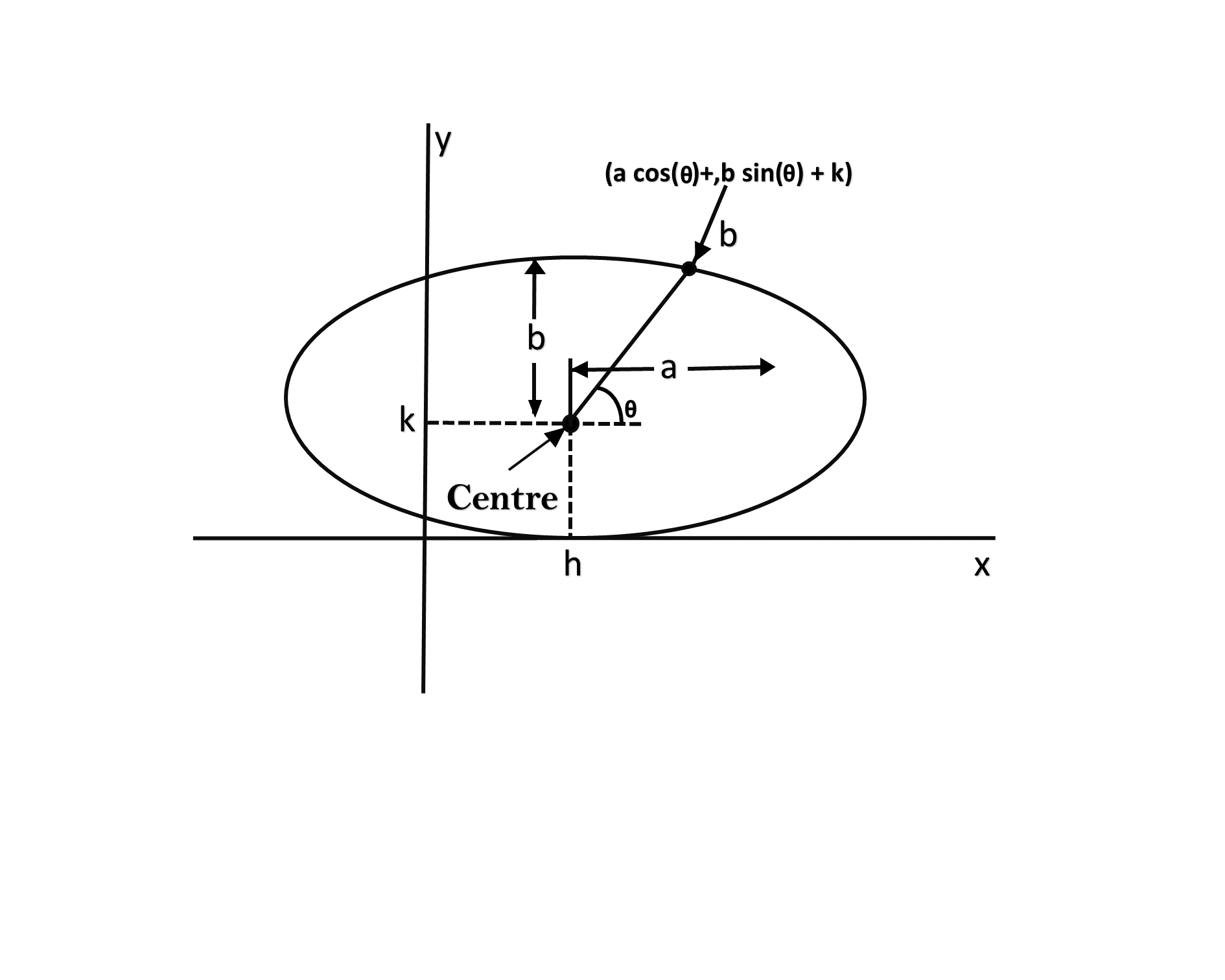


# Trignometric Method:

The following equation defines an ellipse trigonometrically as shown in fig:

x = a \* cos (θ) +h and  
y = b \* sin (θ)+k  
where (x, y) = the current coordinates  
a = length of major axis  
b = length of minor axis  
θ= current angle  
(h, k) = ellipse center

In this method, the value of θ is varied from 0 to Trignometric Methodradians. The remaining points are found by symmetry.



## Drawback:

1. This is an inefficient method.
2. It is not an interactive method for generating ellipse.
3. The table is required to see the trigonometric value.
4. Memory is required to store the value of θ.

## Algorithm:

**Step1:** Start Algorithm

**Step2:** Declare variable x1,y1,aa1,bb1,aa2,bb2,fx,fy,p1,a1,b1

**Step3:** Initialize x1=0 and y1=b/\* values of starting point of circle \*/

**Step4:** Calculate aa1=a1\*a1  
          Calculate bb1=b1\* b1  
          Calculate aa2=aa1\*2  
          Calculate bb2=bb1\*2

**Step5:** Initialize fx = 0

**Step6:** Initialize fy = aa\_2\* b1

**Step7:** Calculate the value of p1and round if it is integer  
          p1=bb1-aa1\* b1+0.25\* aa1/

**Step8:**

While (fx < fy)

{

Set pixel (x1,y1)

Increment x i.e., x = x + 1

Calculate fx = fx + bb2

If (p1 < 0)

Calculate p1 = p1 + fx + bb1/

else

{

Decrement y i.e., y = y-1

Calculate fy = fy - 992;

p1=p1 + fx + bb1-fy

}

}

**Step9:** Setpixel (x1,y1)

**Step10:** Calculate p1=bb1 (x+.5)(x+.5)+aa(y-1)(y-1)-aa1\*bb1

**Step 11:**

While (y1>0)

{

Decrement y i.e., y = y-1

fy=fx-aa2/

if (p1>=0)

p1=p1 - fx + aa1/

else

{

Increment x i.e., x = x + 1

fx= fx+bb\_2

p1=p1+fx-fy-aa1

}

}

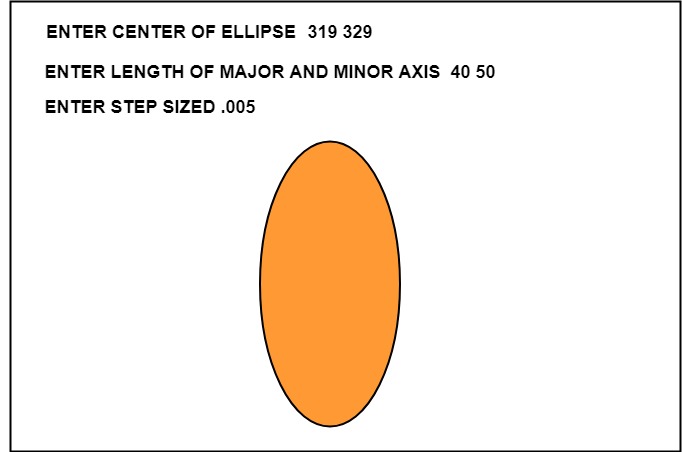
Set pixel (x1,y1)

**Step12:** Stop Algorithm

### Program to draw a Ellipse using Trigonometric method:

1. #include <graphics.h>
2. #include <stdlib.h>
3. #include <math.h>
4. #include <stdio.h>
5. #include <conio.h>
6. #include <iostream.h>
7. # define pi 3.14
9. class bresen
10. {
11. float a, b, h, k, thetaend,step,x,y;
12. int i;
13. public:
14. void get ();
15. void cal ();
16. };
17. void main ()
18. {
19. bresen b;
20. b.get ();
21. b.cal ();
22. getch ();
23. }
24. void bresen :: get ()
25. {
26. cout<<"\n ENTER CENTER OF ELLIPSE";
27. cin>>h>>k;
28. cout<<"\n ENTER LENGTH OF MAJOR AND MINOR AXIS";
29. cin>>a>>b;
30. cout<<"\n ENTER STEP SIZE";
31. cin>> step;
32. }
33. void bresen ::cal ()
34. {
35. /\* request auto detection \*/
36. int gdriver = DETECT,gmode, errorcode;
37. int midx, midy, i;
38. /\* initialize graphics and local variables \*/
39. initgraph (&gdriver, &gmode, " ");
40. /\* read result of initialization \*/
41. errorcode = graphresult ();
42. if (errorcode ! = grOK)    /\*an error occurred \*/
43. {
44. printf("Graphics error: %s \n", grapherrormsg (errorcode);
45. printf ("Press any key to halt:");
46. getch ();
47. exit (1); /\* terminate with an error code \*/
48. }
49. theta= 0;
50. thetaend=(pi\*90)/180;
51. whilex (theta<thetaend)
52. {
53. x = a \* cos (theta);
54. y = b \* sin (theta);
55. putpixel (x+h, y+k, RED);
56. putpixel (-x+h, y+k, RED);
57. putpixel (-x+h, -y+k, RED);
58. putpixel (x+h, -y+k, RED);
59. theta+=step;
60. }
61. getch();
62. }

**Output:**



## Ellipse Axis Rotation:

Since the ellipse shows four-way symmetry, it can easily be rotated. The new equation is found by trading a and b, the values which describe the major and minor axes. When the polynomial method is used, the equations used to describe the ellipse become

Trignometric Method

where (h, k) = ellipse center   
a = length of the major axis  
b = length of the minor axis  
In the trigonometric method, the equations are  
x = b cos (θ)+h       and       y=a sin(θ)+k

Where (x, y) = current coordinates  
a = length of the major axis  
b = length of the minor axis  
θ = current angle  
(h, k) = ellipse center

Assume that you would like to rotate the ellipse through an angle other than 90 degrees. The rotation of the ellipse may be accomplished by rotating the x &y axis α degrees.

          x = a cos (0) - b sin (0+ ∞) + h y= b (sin 0) + a cos (0+∞) + k

