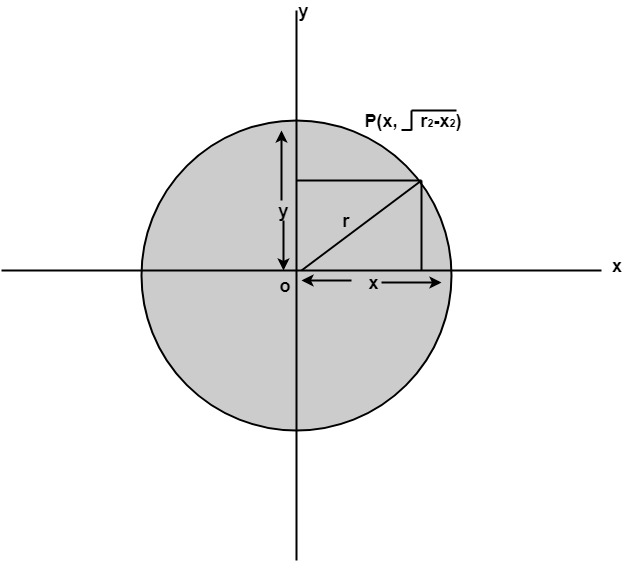
Defining a circle using Polynomial Method:

The first method defines a circle with the second-order polynomial equation as shown in fig:

                    y2=r2-x2  
Where x = the x coordinate  
          y = the y coordinate  
          r = the circle radius

With the method, each x coordinate in the sector, from 90° to 45°, is found by stepping x from 0 to Defining a circle using Polynomial Method & each y coordinate is found by evaluating Defining a circle using Polynomial Method for each step of x.



Algorithm:

**Step1:** Set the initial variables  
          r = circle radius  
          (h, k) = coordinates of circle center  
                x=o  
                I = step size  
                xend= Defining a circle using Polynomial Method

**Step2:** Test to determine whether the entire circle has been scan-converted.

If x > xend then stop.

**Step3:** Compute y = Defining a circle using Polynomial Method

**Step4:** Plot the eight points found by symmetry concerning the center (h, k) 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)  
                Plot (-y + h, x + k)          Plot (y + h, -x + k)  
                Plot (-x + h, y + k)          Plot (x + h, -y + k)

**Step5:** Increment x = x + i

**Step6:** Go to step (ii).

Program to draw a circle using Polynomial Method:

1. #include<graphics.h>
2. #include<conio.h>
3. #include<math.h>
4. voidsetPixel(**int** x, **int** y, **int** h, **int** k)
5. {
6. putpixel(x+h, y+k, RED);
7. putpixel(x+h, -y+k, RED);
8. putpixel(-x+h, -y+k, RED);
9. putpixel(-x+h, y+k, RED);
10. putpixel(y+h, x+k, RED);
11. putpixel(y+h, -x+k, RED);
12. putpixel(-y+h, -x+k, RED);
13. putpixel(-y+h, x+k, RED);
14. }
15. main()
16. {
17. intgd=0, gm,h,k,r;
18. **double** x,y,x2;
19. h=200, k=200, r=100;
20. initgraph(&gd, &gm, "C:\\TC\\BGI ");
21. setbkcolor(WHITE);
22. x=0,y=r;
23. x2 = r/sqrt(2);
24. **while**(x<=x2)
25. {
26. y = sqrt(r\*r - x\*x);
27. setPixel(floor(x), floor(y), h,k);
28. x += 1;
29. }
30. getch();
31. closegraph();
32. **return** 0;
33. }

**Output:**

