The following *pseudo-code* describes the two Bresenham algorithms. The first one is for lines; the second one (on the next page) generates $\frac{1}{8}$ of the circle - the third subroutine draws a full circle by using symmetry. The circle algorithm as written generates the points around the origin (i.e., the (0,0) point). To draw the circle in the proper position, you must “translate” the coordinates - that is, add the (x,y) value of the center to the points on the circle.

```plaintext
procedure BRES_LINE(x1, y1, x2, y2, value: integer);
BEGIN
    dx := ABS(x2 - x1);
    dy := ABS(y2 - y1);
    d := 2*dy - dx;
    incr1 := 2*dy;
    incr2 := 2*(dy - dx);
    if x1 > x2
        then BEGIN
            x := x2;
            y := y2;
            xend := x1;
        END
    else BEGIN
        x := x1;
        y := y1;
        xend := x2;
    END
    WRITE_PIXEL(x, y, value);
    while x < xend do BEGIN
        x := x + 1
        if d < 0
            then d := d + incr1
        else BEGIN
            y := y + 1;
            d := d + incr2;
        END
        WRITE_PIXEL(x, y, value);
    END {while}
END {BRES_LINE}
```
procedure BRES_CIRCLE(radius, value: integer)
    {This routine assumes center of circle is at origin}
BEGIN
    x := 0;
    y := radius;
    d := 3 - 2 * radius;
    while x < y do BEGIN
        CIRCLE_POINTS(x, y, value);
        if d < 0
            then d := d + 4 * x + 6;
        else BEGIN
            d := d + 4 * (x - y) + 10;
            y := y - 1;
        END
        x := x + 1;
    END {while}
    if x = y then CIRCLE_POINTS(x, y, value);
END {BRES_CIRCLE}

procedure CIRCLE_POINTS(x, y, value: integer);
BEGIN
    WRITE_PIXEL(x, y, value);
    WRITE_PIXEL(y, x, value);
    WRITE_PIXEL(y, -x, value);
    WRITE_PIXEL(x, -y, value);
    WRITE_PIXEL(-x, -y, value);
    WRITE_PIXEL(-y, -x, value);
    WRITE_PIXEL(-y, x, value);
    WRITE_PIXEL(-x, y, value);
END {CIRCLE_POINTS}