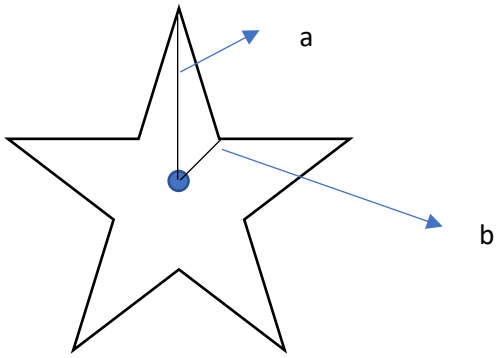
Bài tập giữa kì

1. Vẽ ngôi sao 5 cánh đều:



Thuật toán vẽ Bresenham vẽ đường thẳng với độ dóc m tùy ý:

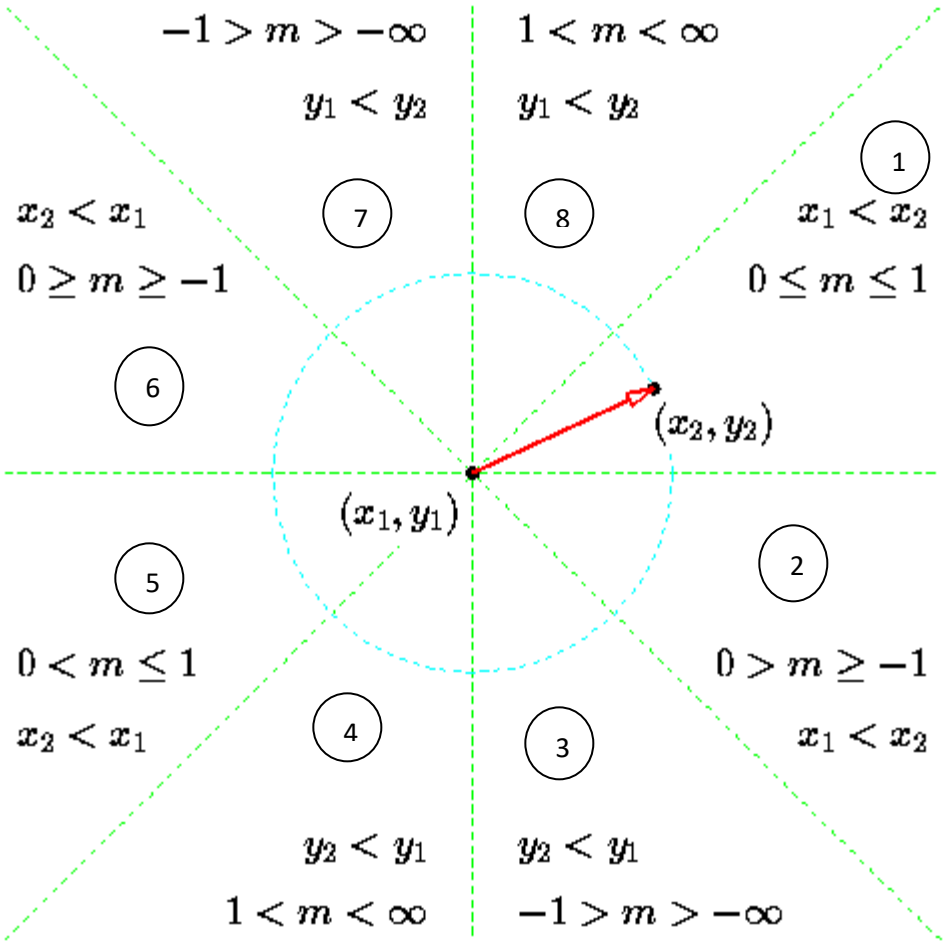
Bài toán: dựa vào tính chất đối xứng trong octant.

Với đường thẳng có m > 1, ta sẽ thay đổi vai trò của các hướng x và y. Đó là ta sẽ tìm giá trị x tiếp theo mà gần với đường thẳng nhất.

Nếu điểm bắt đầu của đường thẳng có độ dóc dương là điểm cuối cùng bên phải, cả x và y đều giảm khi chúng đi từ phải sang trái

Với m < 0, các thủ tục tương tự, khác là 1 tọa độ giảm thì tọa độ khác tăng.

Bảng các phân vùng của octant và các vùng đối xứng: Ví dụ octant 1 đối xứng octant 5



Bảng dưới đây dùng để xác định bộ octant của m:

ΔY	ΔX?ΔY	slope	Octant
≥ 0	≥	Pos ≤ 1	1 (5)
≥ 0	<	Pos >1	2 (6)
< 0	≤	Neg ≥ -1	7 (3)
< 0	>	Neg < -1	8 (4)

Mã giả cho thuật toán Bresenham với m tùy ý:

Function: Bresenham_drawing_line

Inputs: Start point (X1, Y1) , End point (X2, Y2)

Begin

// Điểm đầu

X = X1; Y = Y1;

// Tinh ΔX, ΔY

ΔX = Abs(X2-X1); ΔY = Abs(Y2-Y1);

// Tìm dấu của x2 - x1, y2 – y1

S1 = Sign(X2-X1); S2 = Sign(Y2-Y1);

// Trao đổi vai trò của x và y

If ΔY > ΔX Then

// Swap ΔX và ΔY

T = ΔX; ΔX = ΔY; ΔY = T;

Interchange = 1;

Else

Interchange = 0;

End If

// Tính toán các thông số đầu vào

*P = 2*ΔY - ΔX;*

*A = 2*ΔY;*

*B = 2*ΔY - 2*ΔX*

// Vẽ điểm đầu

Plot(X,Y);

For i = 1 to ΔX Then

If (P < 0) Then

// Nếu Interchange = 1 thì dời Y

// Ngược lại, dời X

If Interchange == 1 Then

```
            Y = Y + S2;
        Else
            X = X + S1;
        End if
        // Cập nhật P
        P = P + A;
    Else // P >= 0
        // Tìm tọa độ X, Y
        Y = Y + S2;     X = X + S1;
        // Cập nhật P
        P = P + B;
    End if
    setPixel(X, Y);
End for
End
```

1.1 Phát biểu bài toán:

Xét đường tròn bán kính a là (C0), đường tròn bán kính b là (C1)

Giới hạn bài toán:

Xét tâm tại (0, 0) tịnh tiến cho ra tâm C tùy ý

Chỉ tính tọa độ 2 điểm đầu: P0 thuộc (C0), P1 thuộc đường (C1). Rồi dùng phép quay điểm P0, P1 1 góc 72 độ cho ra lần lượt các điểm P2 thuộc (C0), P3 thuộc (C1) và cứ thế cho đến khi vẽ hết ngôi sao 5 cánh đều

1.2 Phương pháp:

Giả sử P0(0, a)

Để tính P1(x1, y1) ta dùng công thức sin, cos trong tam giác vuông với góc (a, b) = 36 độ:

x1 = b*sin(36)

y1 = b*cos(36)

Sau đó, ta sẽ dùng thuật toán đường thẳng Bresenham với m tùy ý để kẻ đường thẳng từ

P0 -> P1 để tạo thành cánh của ngôi sao.

Sau đó, ta dùng phép quay để quay điểm P0, P1 theo 1 góc alpha = 72 độ sẽ tạo ra điểm P2, P3 và cứ như thế sẽ tính được hết các điểm của ngôi sao. Công thức phép quay điểm:

x' = cos(alpha) * x – sina * y

y' = sin(alpha) * x + cosa * y

Lưu ý: phép quay với góc alpha dương thường thực hiện theo ngược chiều kim đồng hồ.

Tương tự, khi tính được 2 điểm mới từ phép quay ta sẽ thực hiện nối 2 điểm đó lại với nhau để tạo thành cánh của ngôi sao.

1.3 Giải thuật:

Function: Draw_5point_star
Inputs: Tâm C(xc, yc), Bán kính đường tròn ngoài (a), Bán kính đường tròn trong (b)
Begin

```
// Tinh do doi theo tuc hoanh va truc tung
trX = xc - 0;
trY = yc - 0;

// Tính điểm P0, P1
P0.X = 0;       P0.Y = a;
P1.X = round(b * sin(36 * (PI / 180)));       P1.Y = round(b * cos(36 * (PI / 180)));

// Tinh tien P0
P0 = Translate(P0, trX, trY);
P1 = Translate(P1, trX, trY);

// Kẻ đường thẳng nối P0, P1
Bresenham_drawing_line(P0, P1);
// Khoi báo 2 biến PreviousP, P2, P3
PreviousP = P0, P2 = P0, P3 = P1;
While PreviousP != P0 then // Lay diem P0 lam moc
        // quay điểm P3 trước va tinh tien
        P3 = Rotate(P3, 72);
        P3 = Translate(P3, trX, trY);
        // Nối PreviousP với P3
        Bresenham_drawing_line(PreviousP, P3);
        // quay điểm điểm P2
        P2 = Rotate(P2, 72);
        P2 = Translate(P2, trX, trY);
```

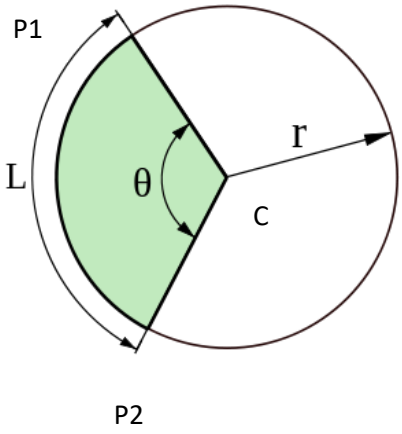
```
            // Nối P3 đến P2
            Bresenham_drawing_line(P3, P2);
            // set lại PreviousP
            PreviousP = P2;
        end while
    End
```

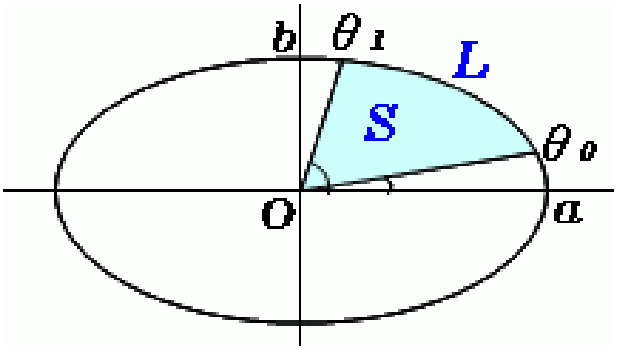
Function: Rotate
Inputs: Điểm quay (P), góc quay (alpha)
Outputs: Trả về điểm đã quay theo góc quay alpha
Begin
Q.x = cos(alpha * *(PI / 180)*) * P.x - sin(alpha * *(PI / 180)*) * P.y;
Q.y = sin(alpha * *(PI / 180)*) * P.x + cos(alpha * *(PI / 180)*) * P.y;
return Q
End

Function: Translate
Inputs: Diem tinh tien (P), Độ dời trục hoành (trX), Độ dời trục tung (trY)
Outputs: Return điểm đã tịnh tiến
Begin
Q.x = P.x + trX;
Q.y = P.y + trY;
return Q
End

2. Vẽ cung tròn/cung ellipse:

Đường tròn: Tâm C, R, P1, P2
Ellipse: Tâm C, Rx, Ry, P1, P2





2.1 Cung tròn:

2.1.1 Phát biểu bài toán:

- Phương trình đường tròn tâm C(xc, yc), bán kính R

    (x – xc)^2 + (y – yc)^2 = R^2

- Vẽ cung tròn trên lưới tọa độ nguyên

- Các điểm vẽ phải thỏa yêu cầu liên tục trong lân cận 8 của điểm ảnh.

- Giới hạn:

    Vẽ tại tâm 0

    Khảo sát cung chắn 1/8

2.1.2 Giải thuật:

// Using modifying Bresenham's circle agorithm

Function: Bresenham_drawing_circular_arc

Inputs: Tâm C(xc, yc), Bán kính R, Điểm P1, Điểm P2

Begin

```
    int x = 0;
    int y = (int)R;
    int p = (int)(5/4 - R);

    // Lay goc dau, goc cuoi cua P1, P2
    // Goc duoc tao boi P1 va 0y, P2 va 0y
    float starting_angle = Calculate_angle_between_P_and_yAsis(P1.x, P1.y, R);
    float ending_angle = Calculate_angle_between_P_and_yAsis(P2.x, P2.y, R);

    // Tìm phân vùng octant bắt đầu và kết thúc của vòng cung (P1, P2)
    int startOctant = findOctantOfAngle(starting_angle);
    int endOctant = findOctantOfAngle(ending_angle);

    Put8Pixel(C.xc, C.yc, x, y, startOctant, endOctant, starting_angle, ending_angle);
```

```
        while (x < y){
                x++;
                if(p < 0)
                        p += 2*x + 3;
                else{
                        y--;
                        p += 2*(x - y) + 5;
                }
                Put8Pixel(C.xc, C.yc, x, y, startOctant, endOctant, starting_angle, ending_angle);
        }
End
```

```
// Hàm put các điểm lấy đối xứng mà có nằm trong vòng cung (P1, P2)
Function: Put8Pixel
Inputs: Tâm C(xc, yc), Hoành độ x, Tung độ y, Phân vùng bắt đầu startOctant,
Phân vùng kết thúc endOctant, Goc bắt đầu starting_angle, Góc kết thúc ending_angle
Begin
        // Duyệt từ startOctant đến endOctant để tiết kiệm thời gian
        // Ý tưởng: mỗi lần duyệt đến Octant nào thì sẽ lấy điểm đối xứng cho octant đó
        // và kiểm tra xem điểm đối xứng đó có nằm trong vòng cung (P1, P2) hay không?
        // Nếu có thì putpixel điểm đó vào
        for (int i = startOctant; i <= endOctant; i++){
                switch(i){
                        case 0:
                                if(isInsideArc(x, y, starting_angle, ending_angle))
                                        putpixel(xc + x, yc + y);
                                break;
```

```
            case 1:
                if(isInsideArc(y, x, starting_angle, ending_angle))
                    putpixel(xc + y, yc + x);
                break;
            case 2:
                if(isInsideArc(y, -x, starting_angle, ending_angle))
                    putpixel(xc + y, yc-x);
                break;
            case 3:
                if(isInsideArc(x, -y, starting_angle, ending_angle))
                    putpixel(xc + x, yc-y);
                break;
            case 4:
                if(isInsideArc(-x, -y, starting_angle, ending_angle))
                    putpixel(xc-x, yc-y);
                break;
            case 5:
                if(isInsideArc(-y, -x, starting_angle, ending_angle))
                    putpixel(xc-y, yc-x);
                break;
            case 6:
                if(isInsideArc(-y, x, starting_angle, ending_angle))
                    putpixel(xc-y, yc+x);
                break;
            default:
                if(isInsideArc(-x, y, starting_angle, ending_angle))
                    putpixel(xc-x, yc+y);
        }
```

```
        }
End

// Hàm kiểm tra 1 điểm có nằm trong vòng cung hay không
Function: isInsideArc
Inputs: Toa độ x, y, Góc bắt đầu starting_angle, Góc kết thúc ending_angle
Begin
        // Tính góc tạo bởi P và Oy
        alpha = Calculate_angle_between_P_and_yAsis(P, R)
        if (starting_angle <= alpha && alpha <= ending_angle)
                return 1;
        return 0;
End

// Hàm kiểm tra 1 góc nằm trong phân vùng nào của phân vùng 8 (Octant)
// Các phân vùng trong octant đánh số từ 0 - 7 theo chiều kim đồng hồ
Function: findOctantOfAngle
Inputs: Góc angle
Begin
        return int(angle / (PI/4));
End

// Hàm để tính góc tạo bởi điểm P và trục Oy.
Function: Calculate_angle_between_P_and_yAsis
Inputs: Diem (x, y), Ban kinh R
Begin
        // Tinh theo radian
```

```
        // Một hình tròn được chia làm 4 phân vùng (quadrant)
        // Vùng 0: 0 - PI/2
        // Vùng 1: PI/2 - PI
        // Vùng 2: PI - 3*PI / 2
        // Vùng 3: 3*PI/2 - 2*PI

        if (((0 <= x && x < R) && (0 < y && y <= R))) // Vung 0
                return arccos(y / R);
        else if ((0 < x && x <= r) && (-r < y && y <= 0)) // Vung 1
                return arcsin(y / R) + PI/2;
        else if ((-r < x && x <= 0) && (-r <= y && y < 0)) // Vung 2
                return arccos(y / R) + PI;
        else // Vung 3
                return arcsin(y / R) + (3*PI)/2;
End
```

2.2 Cung ellipse:

Giải thuật:

```
// Using modifying Bresenham's ellipse agorithm
Function: Bresenham_drawing_ellipse_arc
Inputs: Tâm C(xc, yc), Bán kính (Rx, Ry), Điểm P1, Điểm P2
Begin
        // Điểm đầu
        int x = 0;
        int y = (int)Ry;

        float p = Ry*Ry - Rx*Rx*Ry + (1/4)*Rx*Rx;
        float A = 2*Ry*Ry*x;
        float B = 2*Rx*Rx*y;
```

```
// Lay goc dau, goc cuoi cua P1, P2
// Goc duoc tao boi P1 va 0y, P2 va 0y
float starting_angle = Calculate_angle_between_P_and_yAsis(P1.x, P1.y, Ry);
float ending_angle = Calculate_angle_between_P_and_yAsis(P2.x, P2.y, Ry);

// Tìm phân vùng Quadrant bắt đầu và kết thúc của vòng cung (P1, P2)
int startQuadrant = findQuadrantOfAngle(starting_angle);
int endQuadrant = findQuadrantOfAngle(ending_angle);

// Ve diem dau
Put4Pixel(C.xc, C.yc, x, y, startQuadrant, endQuadrant, starting_angle, ending_angle);
// Xét vùng 1: 0 < |dy/dx| <= 1
int k = 0;
while (2*Ry*Ry < 2*Rx*Rx*y){
        x++;
        if(p < 0){
                A +=  2*Ry*Ry;
                p += A + Ry*Ry;
        }
        else{
                y--;
                A += 2*Ry*Ry;
                B -= 2*Rx*Rx;
                p += A - B + Ry*Ry;
        }
        Put4Pixel(C.xc, C.yc, x, y, startQuadrant, endQuadrant, starting_angle,
ending_angle);
}
```

```
        // Xét vùng 2: |dy/dx| > 1
        float xlast = x, ylast = y;
        A = 2*Ry*Ry*xlast;
        B = 2*Rx*Rx*ylast;
        p = Ry*Ry*(xlast + 1/2)^2 + Rx*Rx*(ylast - 1)^2 - Rx*Rx*Ry*Ry;

        k = 0;
        while(y != 0){
                y--;
                if(p < 0){
                        x++;
                        A += 2*Ry*Ry;
                        B -= 2*Rx*Rx;
                        p += A - B + Rx*Rx;
                }
                else{
                        B -= 2*Rx*Rx;
                        p -= B + Rx*Rx;
                }
                Put4Pixel(C.xc, C.yc, x, y, startQuadrant, endQuadrant, starting_angle,
ending_angle);
        }
End

// Hàm put các điểm lấy đối xứng mà có nằm trong vòng cung (P1, P2)
Function: Put4Pixel
Inputs: Tâm C(xc, yc), Hoành độ x, Tung độ y, Phân vùng bắt đầu startQuadrant,
Phân vùng kết thúc endQuadrant, Goc bắt đầu starting_angle, Góc kết thúc ending_angle
Begin
```

```
        // Duyệt từ startQuadrant đến endQuadrant để tiết kiệm thời gian
        // Ý tưởng: mỗi lần duyệt đến Quadrant nào thì sẽ lấy điểm đối xứng cho Quadrant đó
        // và kiểm tra xem điểm đối xứng đó có nằm trong vòng cung (P1, P2) hay không?
        // Nếu có thì putpixel điểm đó vào
        for (int i = startQuadrant; i <= endQuadrant; i++){
                switch(i){
                        case 0:
                                if(isInsideArc(x, y, starting_angle, ending_angle))
                                        putpixel(xc+x, yc+y);
                                break;
                        case 1:
                                if(isInsideArc(-y, x, starting_angle, ending_angle))
                                        putpixel(xc-y, yc+x);
                                break;
                        case 2:
                                if(isInsideArc(-x, -y, starting_angle, ending_angle))
                                        putpixel(xc-x, yc-y);
                                break;
                        case 3:
                                if(isInsideArc(-x, y, starting_angle, ending_angle))
                                        putpixel(xc-x, yc+y);
                                break;
                }
        }
End

// Hàm kiểm tra 1 điểm có nằm trong vòng cung hay không
Function: isInsideArc
```

```
Inputs: Toa độ x, y, Góc bắt đầu starting_angle, Góc kết thúc ending_angle
Begin
	// Tính góc tạo bởi P và Oy
	alpha = Calculate_angle_between_P_and_yAsis(P, R)
	if (starting_angle <= alpha && alpha <= ending_angle)
		return 1;
	return 0;
End

// Hàm kiểm tra 1 góc nằm trong phân vùng nào của phân vùng 4 (Quadrant)
// Các phân vùng trong Quadrant đánh số từ 0 - 3 theo chiều kim đồng hồ
Function: findQuadrantOfAngle
Inputs: Góc angle
Begin
	return int(angle / (PI/2));
End

// Hàm để tính góc tạo bởi điểm P và trục Oy.
Function: Calculate_angle_between_P_and_yAsis
Inputs: Diem (x, y), Ban kinh R
Begin
	// Tinh theo radian

	// Một hình tròn được chia làm 4 phân vùng (quadrant)
	// Vùng 0: 0 - PI/2
	// Vùng 1: PI/2 - PI
	// Vùng 2: PI - 3*PI / 2
	// Vùng 3: 3*PI/2 - 2*PI
```

```
        if (((0 <= x && x < R) && (0 < y && y <= R))) // Vung 0
                return arccos(y / R);
        else if ((0 < x && x <= r) && (-r < y && y <= 0)) // Vung 1
                return arcsin(y / R) + PI/2;
        else if ((-r < x && x <= 0) && (-r <= y && y < 0)) // Vung 2
                return arccos(y / R) + PI;
        else // Vung 3
                return arcsin(y / R) + (3*PI)/2;
End
```

3. Vẽ đường cong Bezier bằng phương pháp de Casteljau với điểm chia theo tỷ lệ 1/2:

```
Point Casteljau(float t)
{
        Point Q[Max];
        int i, r;
        for (i = 0; i <= L; i++)
        {
                Q[i].x = P[i].x;
                Q[i].y = P[i].y;
        }
        for (r = 1 ; r <= L; r++)
        {
                for (i = 0; i <= L - r; i++)
                {
                        Q[i].x = (1 - t)*Q[i].x + t*Q[i + 1].x;
                        Q[i].y = (1 - t)*Q[i].y + t*Q[i + 1].y;
                }
```

```
        }
        return(Q[0]);
}

// Để vẽ đường cong Bezier xem Casteljau là 1 thủ tục phụ trong DrawCurve
void DrawCurve(float a, float b, int NumPoints)
{
        float Delta = (b – a)/(float)NumPoints;
        float t = a;
        int i;
        moveto(Casteljau(t).x, Casteljau(t).y) ;
        for (i = 1; i <= NumPoints; i++)
        {
                t += Delta ;
                lineto(Casteljau(t).x, Casteljau(t).y) ;
        }
}
```

4. Xác định số viên gạch kích thước cxd (mm) cần để nền hồ Ellipse (Bán trục lớn: a (cm), Bán trục nhỏ: b (cm) hoặc Circle (Bán kính: R):

4.1 Đường tròn:

```
// Using modifying Bresenham's circle agorithm
Function: Count_brick
Inputs: Tâm C(xc, yc), Bán kính R, Điểm P1, Điểm P2
Begin
    int x = 0;
    int y = (int)R;
    int p = (int)(5/4 - R);
    int count = 0;
```

```
        Put8Pixel(C.xc, C.yc, x, y);
        while (x < y){
                x++;
                if(p < 0)
                        p += 2*x + 3;
                else{
                        y--;
                        p += 2*(x - y) + 5;
                }
                Put8Pixel(C.xc, C.yc, x, y);
        }

        // Goi ham to mau de tinh so gach nen
        // Loang tu tam
        FloodFill(xc, yc, 1, 0, count);

End

// Hàm put các điểm lấy đối xứng
Function: Put8Pixel
Inputs: Tâm C(xc, yc), Hoành độ x, Tung độ y
Begin
        putpixel(xc+x, yc+y);
        putpixel(xc+y, yc+x);
        putpixel(xc+y, yc-x);
        putpixel(xc+x, yc-y);
        putpixel(xc-x, yc-y);
        putpixel(xc-y, yc-x);
        putpixel(xc-y, yc+x);
```

```
        putpixel(xc-x, yc+y);
    End

    // Hàm tô màu
    void FloodFill(int x, int y, int fill_color, int boundary, int count = 0){
        if (getpixel(x, y) != fill_color && getpixel(x, y) != boundary){
                putpixel(x, y, fill_color);
                count++;
                FloodFill(x+1, y, fill_color, boundary);
                FloodFill(x-1, y, fill_color, boundary);
                FloodFill(x, y+1, fill_color, boundary);
                FloodFill(x, y-1, fill_color, boundary);
        }
    }

    void ScanLine(int l, int t, int r, int b, int fill_color, int count = 0){
        setcolor(fill_color);
        for(int y = t; y < b; y++){
                line(l, y, r, r);
                // Xử lý số gạch ở đây

        }
    }
```

4.2 Ellipse:

```
// Using modifying Bresenham's ellipse agorithm
Function: Count_brick_ellipse_pool
Inputs: Tâm C(xc, yc), Bán kính (Rx, Ry)
Begin
    // Điểm đầu
```

```
int x = 0;
int y = (int)Ry;

float p = Ry*Ry - Rx*Rx*Ry + (1/4)*Rx*Rx;
float A = 2*Ry*Ry*x;
float B = 2*Rx*Rx*y;

int count = 0;

// Ve diem dau
Put4Pixel(C.xc, C.yc, x, y);
// Xét vùng 1: 0 < |dy/dx| <= 1
int k = 0;
while (2*Ry*Ry < 2*Rx*Rx*y){
        x++;
        if(p < 0){
                A +=  2*Ry*Ry;
                p += A + Ry*Ry;
        }
        else{
                y--;
                A += 2*Ry*Ry;
                B -= 2*Rx*Rx;
                p += A - B + Ry*Ry;
        }
        Put4Pixel(C.xc, C.yc, x, y);
}
// Xét vùng 2: |dy/dx| > 1
float xlast = x, ylast = y;
```

```
A = 2*Ry*Ry*xlast;
B = 2*Rx*Rx*ylast;
p = Ry*Ry*(xlast + 1/2)^2 + Rx*Rx*(ylast - 1)^2 - Rx*Rx*Ry*Ry;

k = 0;
while(y != 0){
        y--;
        if(p < 0){
                x++;
                A += 2*Ry*Ry;
                B -= 2*Rx*Rx;
                p += A - B + Rx*Rx;
        }
        else{
                B -= 2*Rx*Rx;
                p -= B + Rx*Rx;
        }
        Put4Pixel(C.xc, C.yc, x, y);
}

// Goi ham to mau de dem gach
//...


End

// Hàm put các điểm lấy đối xứng mà có nằm trong vòng cung (P1, P2)
Function: Put4Pixel
Inputs: Tâm C(xc, yc), Hoành độ x, Tung độ y
```

```
Begin
    putpixel(xc+x, yc+y);
    putpixel(xc-y, yc+x);
    putpixel(xc-x, yc-y);
    putpixel(xc-x, yc+y);
End

void ScanLine(int l, int t, int r, int b, int fill_color, int count = 0){
    setcolor(fill_color);
    for(int y = t; y < b; y++){
            line(l, y, r, r);
            // Xử lý số gạch ở đây

    }
}
```