STRUCTURA SCRIPTULUI

1.Incarca imaginea si o converteste in gri

```
% 1. Încarcă imaginea
img  = imread('1305B_1306A.png');
imgG = rgb2gray(img);
```

– Grayscale-ul este mai ușor de binarizat și procesat decât imaginea RGB.

2. Aplică binarizare automata si inverseaza

```
% 2. Mască Otsu inversată
level = graythresh(imgG);
mask  = imcomplement(imbinarize(imgG, level));
```

graythresh: determină un prag automat de separare

imbinarize: convertește în alb/negru.

imcomplement: inversează masca (fundal devine negru, formele albe).

3. Găsește componentele conexe și le analizează

```
% 3. Contururi + proprietăți
[B,L]   = bwboundaries(mask,'noholes');
stats   = regionprops(L, ...
    'BoundingBox','Centroid','Area','Image','EulerNumber','Circularity');
```

-bwboundaries: extrage contururile.
-regionprops extrage:

- BoundingBox: dreptunghiul de încadrare

- Centroid: centrul
- Area: suprafață
- Image: masca fiecărei forme
- EulerNumber: pentru găuri (ex. triunghi cu gaură)
- Circularity: pt cerc/elipsă (ideal 1 pentru cerc)

4. Sortează regiunile după dimensiune

```
% 4. Sortează descrescător după arie
areas = [stats.Area];
[~, idx] = sort(areas,'descend');
```

– Se sortează descrescător în funcție de suprafață. idx(1) = cea mai mare formă.

5. Definește nume pentru primele 7 forme

```
% 5. Denumiri presetate pentru primele 7 regiuni
shapeNames = { ...
    'dreptunghi',                    ... % idx(1)
    'elipsă',                        ... % idx(2)
    'stea',                          ... % idx(3)
    'cerc',                          ... % idx(4)
    'triunghi cu vârful in jos',     ... % idx(5)
    'triunghi cu varful in sus',     ... % idx(6)
    'triunghi cu gaura'                  % idx(7)
};
```

6. Parcurge primele 8 regiuni și le clasifică

```
% 6. Afișează primele 8 regiuni și detectează „patrat rotit”
figure; imshow(img); hold on;
for k = 1:8
    i    = idx(k);
    bb   = stats(i).BoundingBox;
    ctr  = stats(i).Centroid;
    reg  = stats(i).Image;
    eul  = stats(i).EulerNumber;
    cir  = stats(i).Circularity;

    % umple golurile + calculează orientarea și numărul de colțuri
    regF = imfill(reg,'holes');
    ort  = regionprops(regF,'Orientation').Orientation;
    nc   = numel(corner(regF,'QualityLevel',0.1,'SensitivityFactor',0.2));
```

7. Calculează raportul laturilor

```
% raport laturi
rap = bb(3)/bb(4);
```

8. Reguli de suprascriere pentru formele cu 4 colțuri

```
% clasificare cu suprascrieri:
if nc == 4 && abs(rap-1)<0.2 && abs(abs(ort)-45)<15
    label = 'romb';                 % diamant cyan
elseif nc == 4 && abs(rap-1)<0.2
    label = 'patrat rotit';         % patrat galben
else
```

9. Dacă nu e una specială, folosește din shapeNames

```
    % pentru primele 7, folosim numele prestabilite
    if k <= numel(shapeNames)
        label = shapeNames{k};
    else
        label = '';  % opțional, nimic pentru k>7
    end
end
```

10. Desenează chenarul + textul dacă s-a găsit o etichetă

```
    % desenează doar dacă avem o etichetă
    if ~isempty(label)
        rectangle('Position',bb,'EdgeColor','g','LineWidth',2);
        text( ctr(1), ctr(2), label, ...
              'HorizontalAlignment','center', ...
              'VerticalAlignment','middle', ...
              'FontSize',12, 'Color','w', 'FontWeight','bold');
    end
end
hold off;
```

Rezultat final :