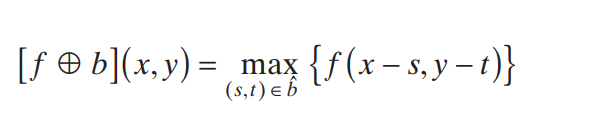
**Class Lab 8**

Use two images for each operation to do the following operations and write down their advantages and disadvantages and explain your results:

1. **Dilation (noisy\_rectangle, noisy\_fingerprint):**

**Algorithm:**



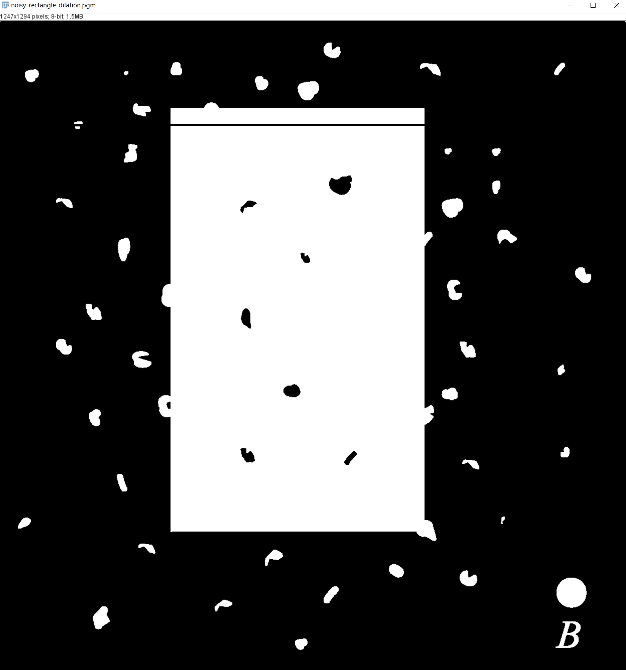
**Results (including pictures):**

Result of processing “noisy\_rectangle.pgm”:

Source Image:

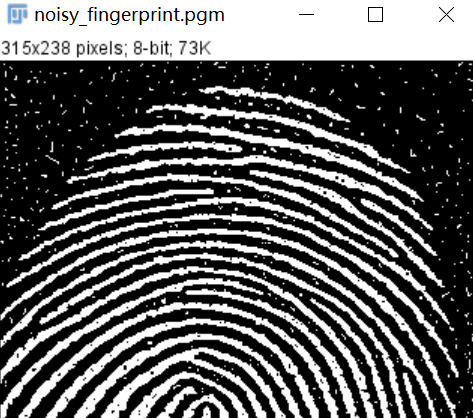


Result after dilation:

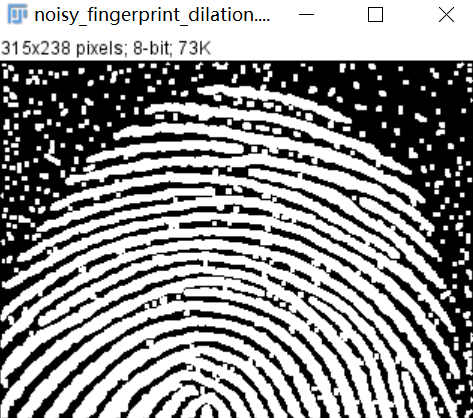


Result of processing “noisy\_fingerprint.pgm”:

Source Image:



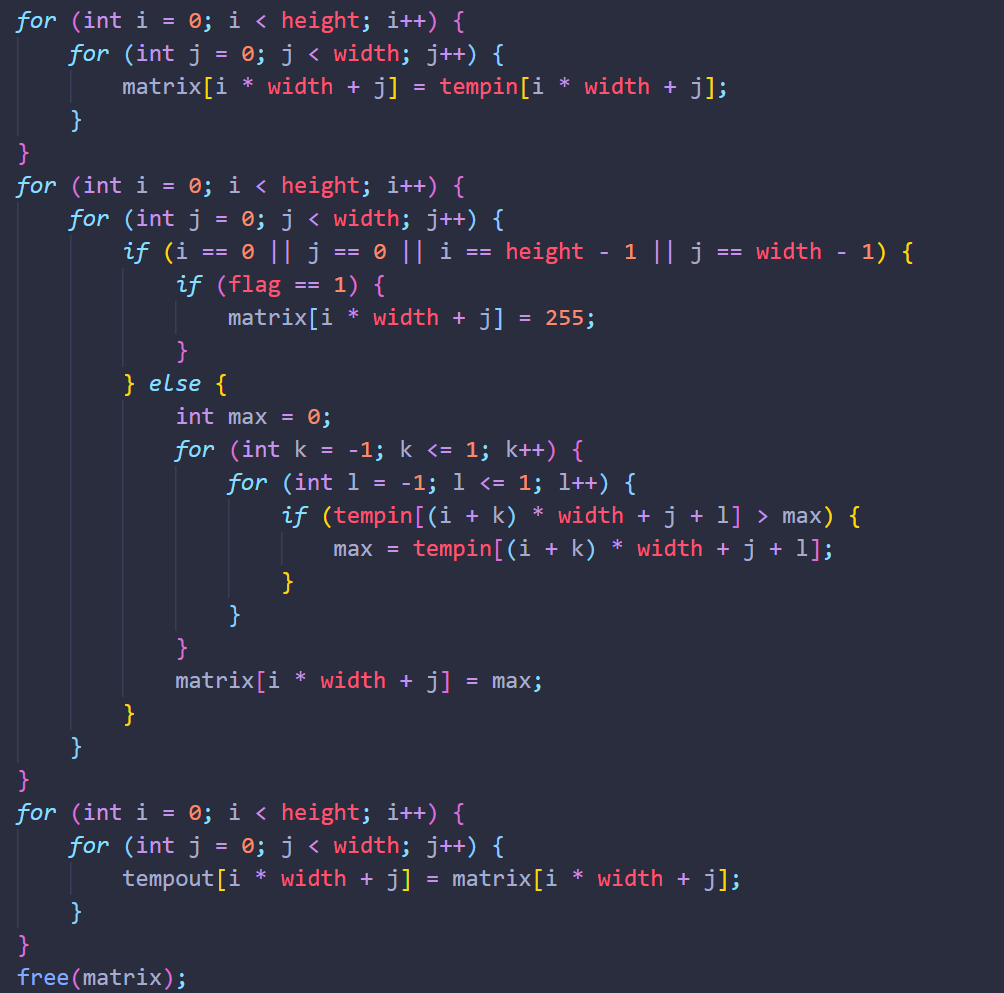
Result after dilation dilation:



**Discussion:**

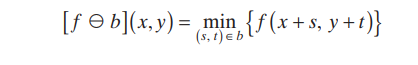
Dilation “grows” or “thickens” objects in a binary image. The manner and extent of this thickening is controlled by the shape and size of the structuring element used.

**Codes:**



1. **EROSION (noisy\_rectangle, noisy\_fingerprint):**

**Algorithm:**



**Results (including pictures):**

Result of processing “noisy\_rectangle.pgm”:

Source Image:

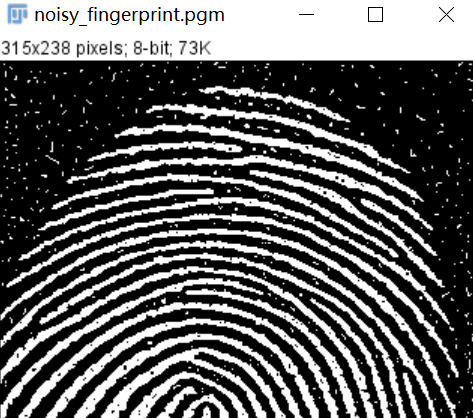


Result after EROSION



Result of processing “noisy\_fingerprint.pgm”:

Source Image:



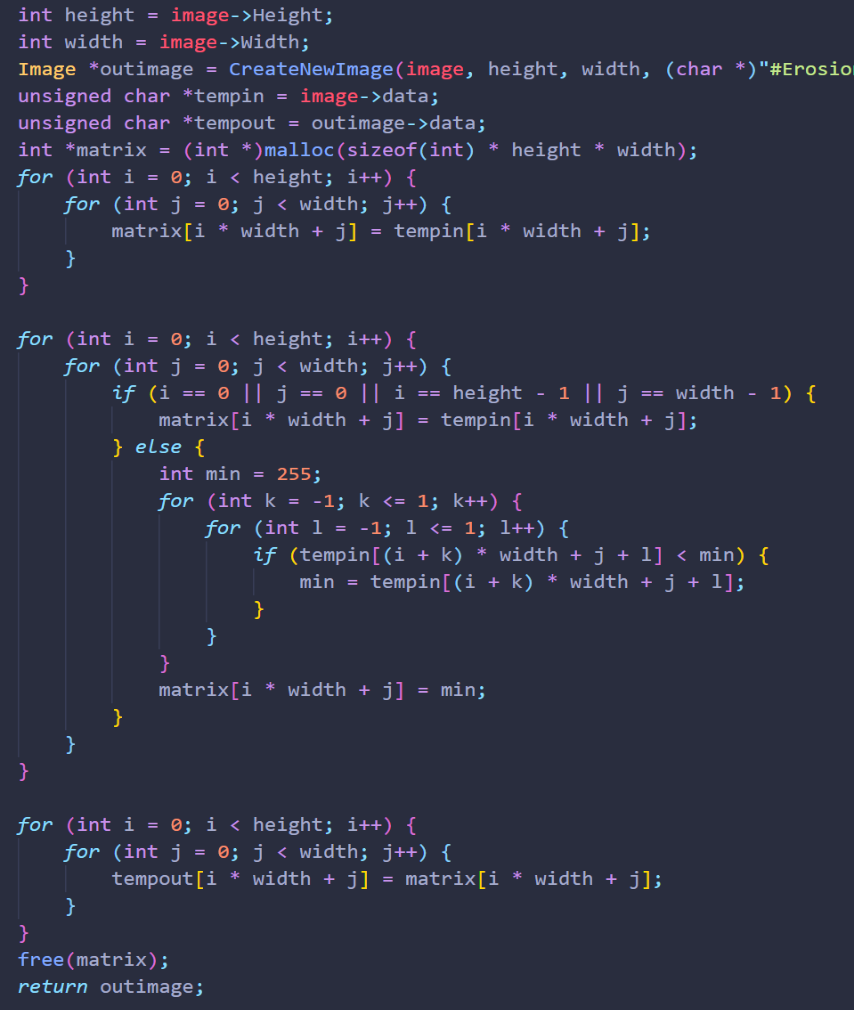
Result after EROSION:



**Discussion:**

Because grayscale erosion with a flat SE computes the minimum intensity value of f in every neighborhood of (,) x y coincident with b, we expect in general that an eroded grayscale image will be darker than the original, that the sizes (with respect to the size of the SE) of bright features will be reduced, and that the sizes of dark features will be increased.

**Codes:**



1. **Opening (noisy\_rectangle, noisy\_fingerprint):**

**Algorithm:**

1. Erosion
2. Dilation

**Results (including pictures):**

Result of processing “noisy\_rectangle.pgm”:

Source Image:

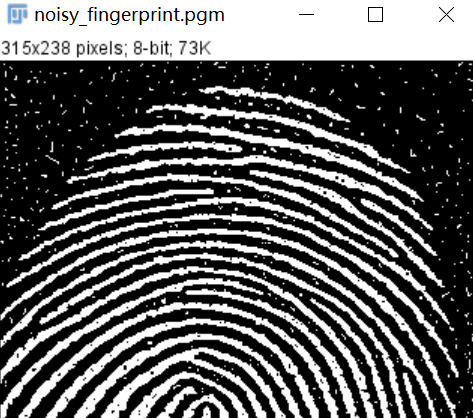


Result after Opening

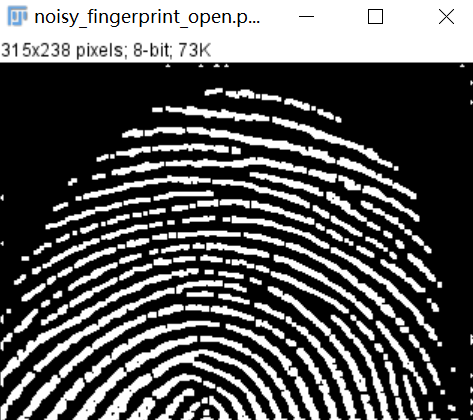


Result of processing “noisy\_fingerprint.pgm”:

Source Image:



Result after Opening:



**Discussion:**

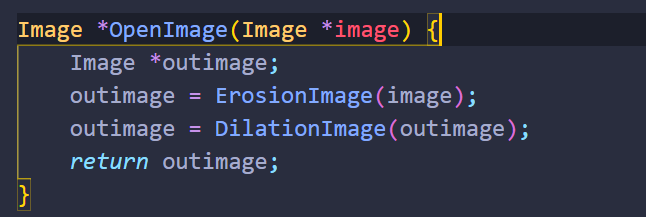
(1) The open operation can remove isolated small points, burrs and small bridges, and the overall position and shape are inconvenient.

(2) Open operation is a filter based on geometric operation.

(3) Different sizes of structuring elements will lead to different filtering effects.

(4) The selection of different structural elements leads to different segmentations, that is, different features are extracted.

**Codes:**



1. **Closing (noisy\_rectangle, noisy\_fingerprint):**

**Algorithm:**

1. Dilation
2. Erosion

**Results (including pictures):**

Result of processing “noisy\_rectangle.pgm”:

Source Image:

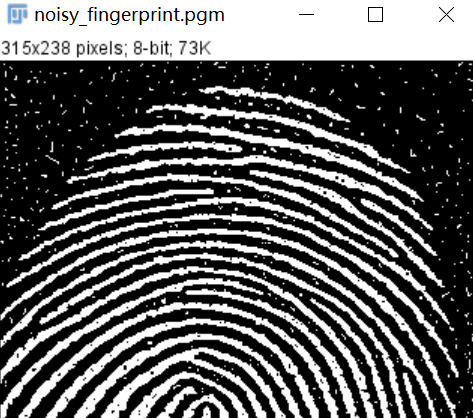


Result after Closing

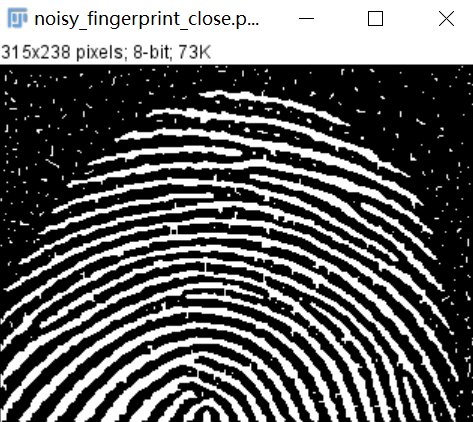


Result of processing “noisy\_fingerprint.pgm”:

Source Image:



Result after Closing:



**Discussion:**

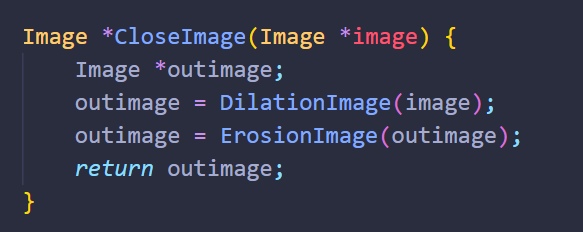
(1) The closing operation can fill in small lakes (ie small holes) and bridge small cracks, while the overall position and shape remain unchanged.

(2) The closing operation filters the image by filling the concave corners of the image.

(3) Different sizes of structuring elements will lead to different filtering effects.

(4) The choice of different structural elements leads to different segmentations.

**Codes:**



1. **Extract the Boundaries (licoln, U):**

**Algorithm:**

1. Origin image – Erosion image

**Results (including pictures):**

Result of processing “licoln.pgm”:

Source Image:

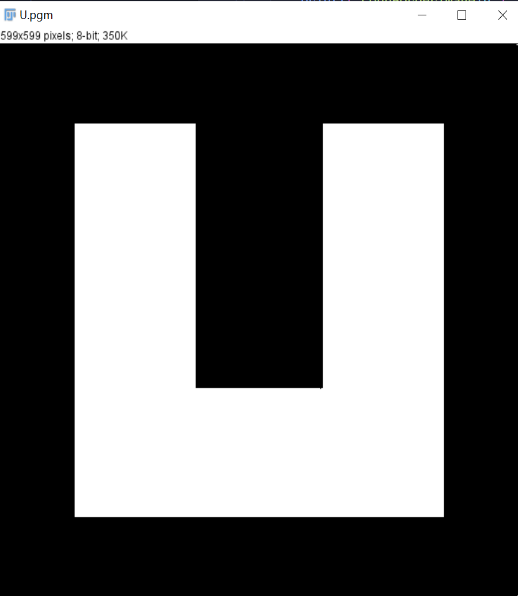


Result after Extract the Boundaries

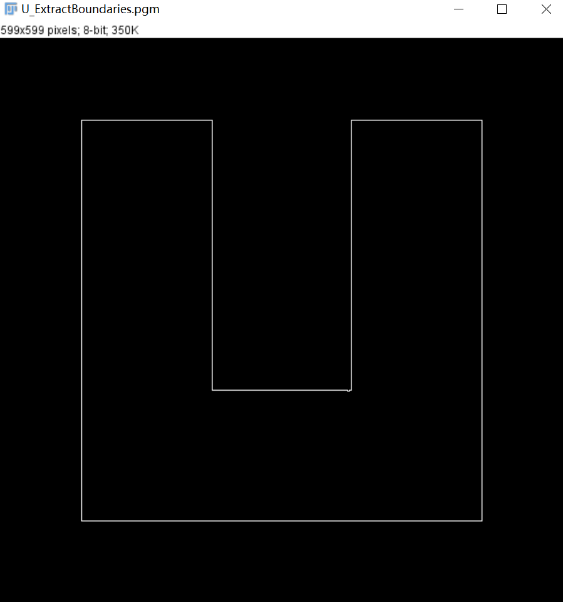


Result of processing “U.pgm”:

Source Image:



Result after Extract the Boundaries:



**Discussion:**

When a 3x3 1-value struct is used, the extracted boundary value is 1 pixel.

**Codes:**



1. **Count the number of pixels in each white connected component (licoln, U):**

**Algorithm:**

Suppose the original image is A0, the iteration matrix is X1, and B is the connected component storage matrix

1. A=A0

2. Find the point in A where the gray value is 1

3. Select the first point with a gray value of 1 as the initial point

4. Build a 3x3 1-value struct

5. Find the intersection after dilation

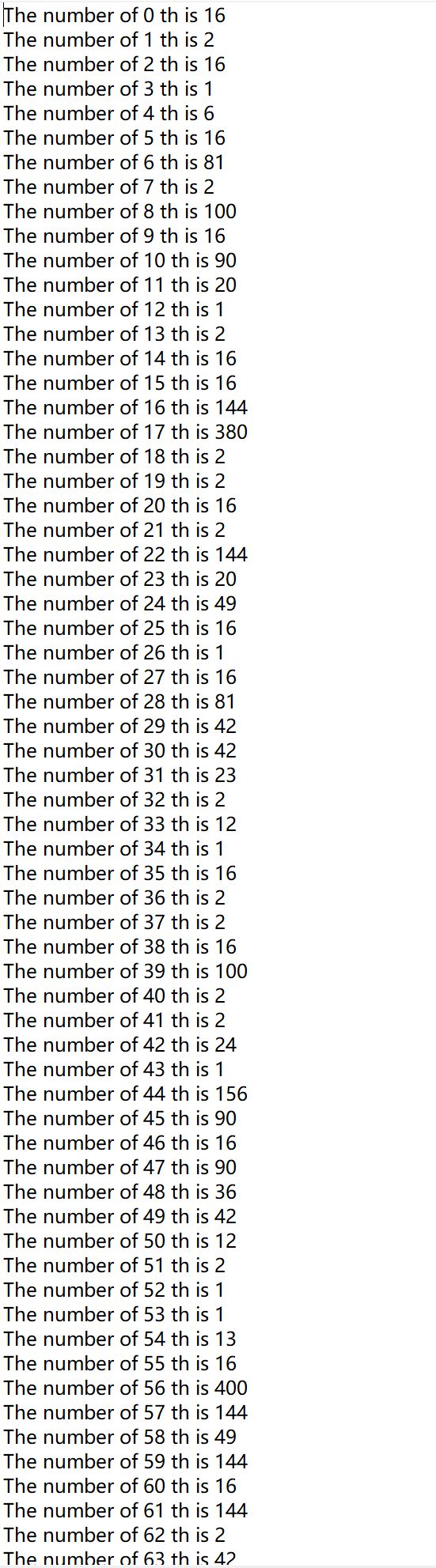
6. If X1 == Xp1, the iteration ends

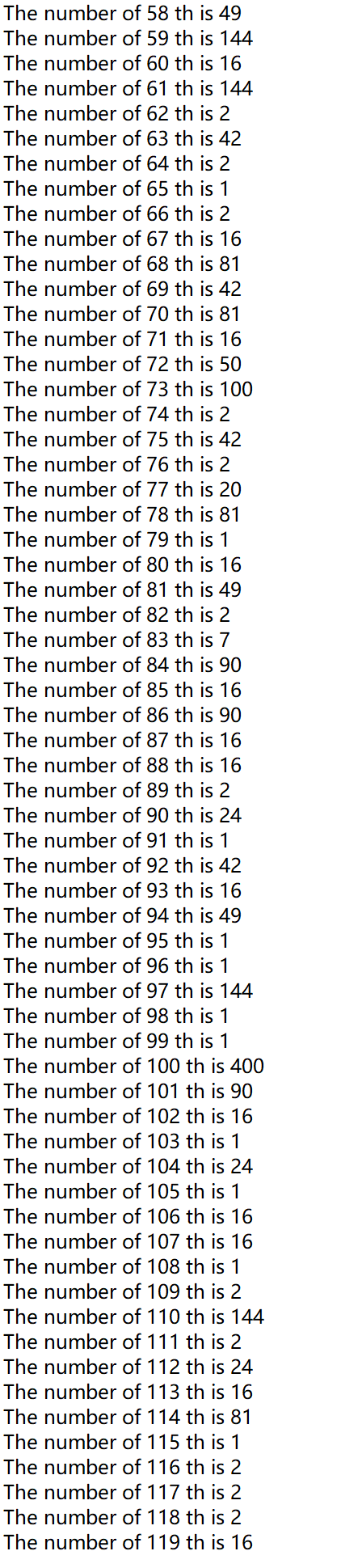
7. B = B+X1

8. A = A-B

Continue looping until there are no white dots in A

**Results (including pictures**

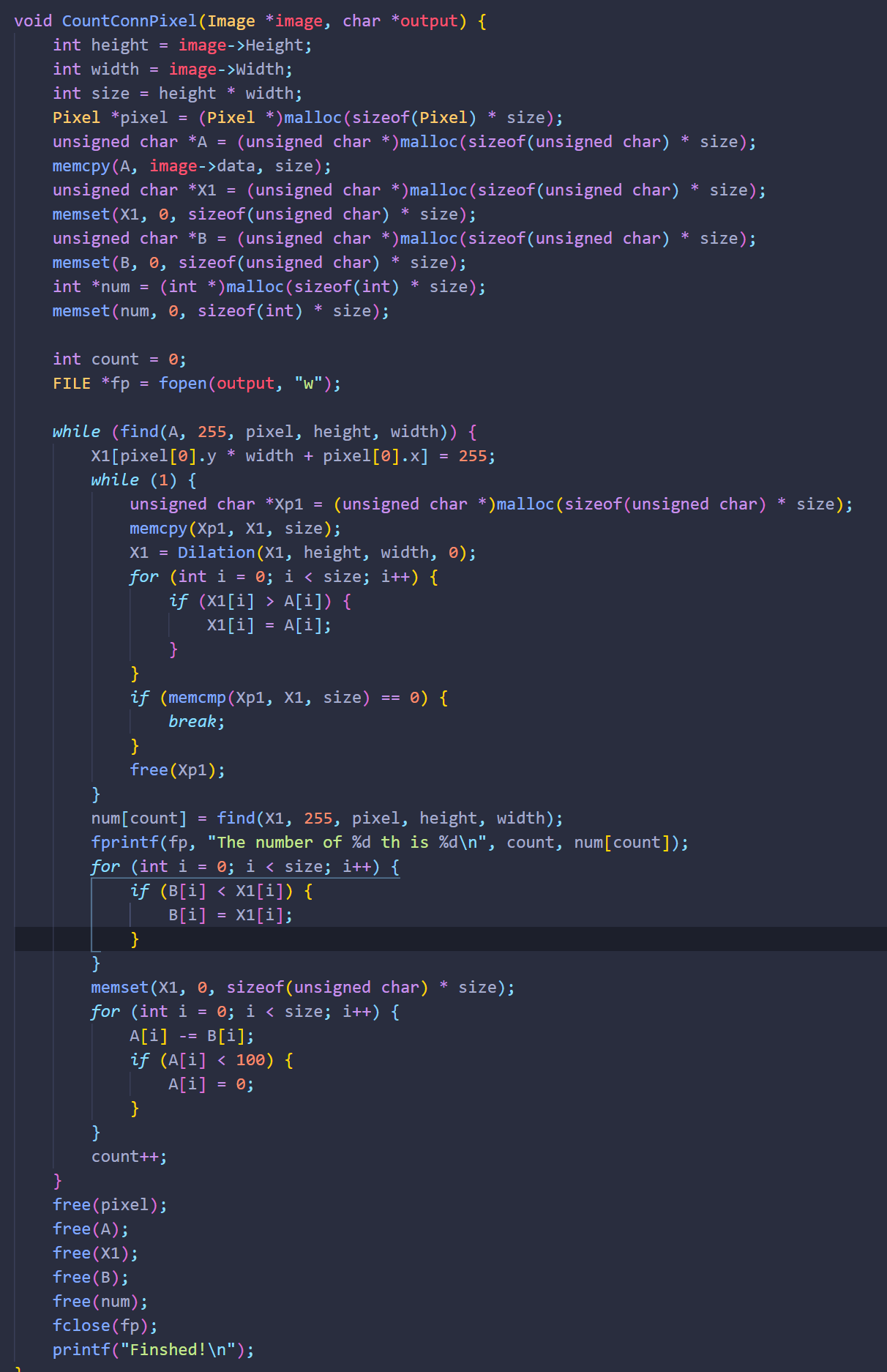




**Discussion:**

Repeated iteration, the efficiency is too low

**Codes:**



1. **Particles that only coincide with the edges of the image (**bubbles\_on\_black\_background**):**

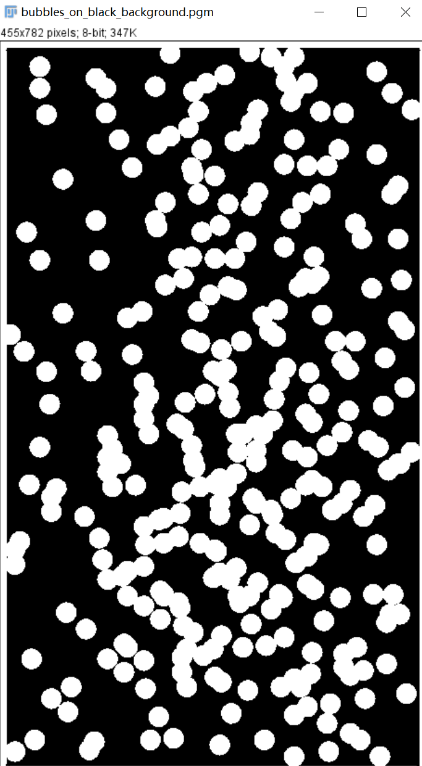
**Algorithm:**

Color the image border pixels the same color as the particles (white). Call the resulting set of border pixels B. Apply the connected component algorithm. All connected components that contain elements from B are particles that have merged with the border of the image

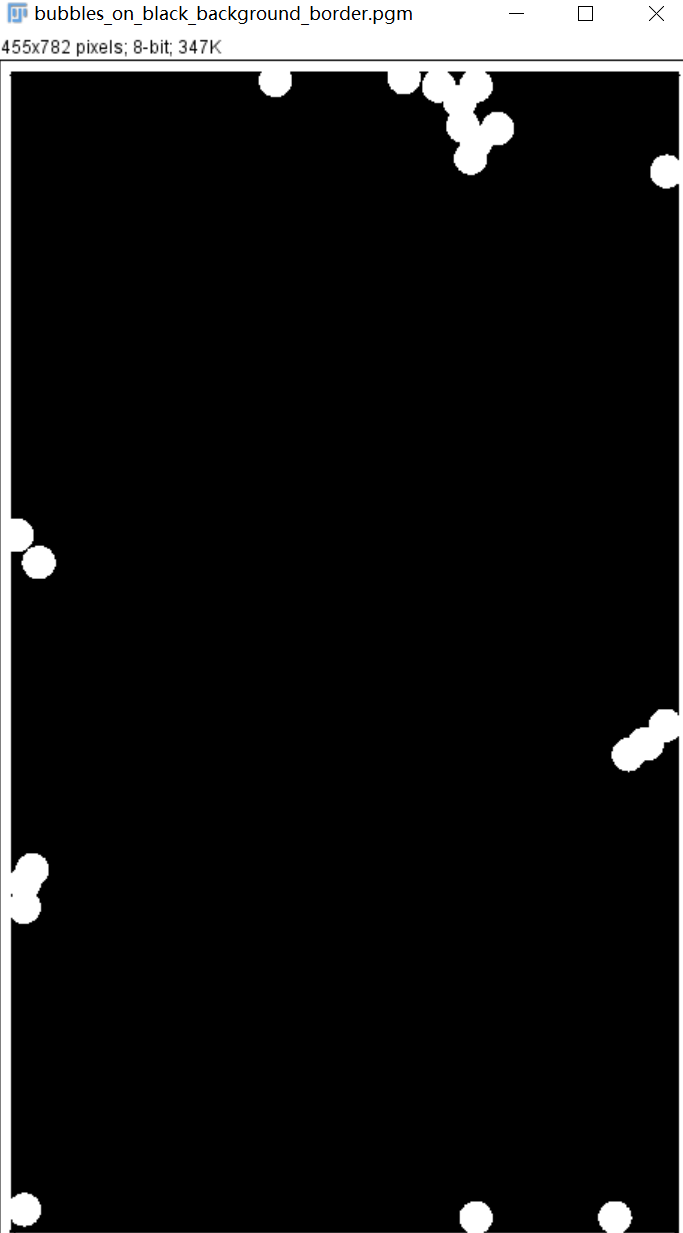
**Results (including pictures):**

Result of processing “bubbles\_on\_black\_background.pgm”:

Source Image:



Result：



**Discussion:**

Extraction of connected components for a fully automated process

**Codes:**



1. **Only particles that overlap each other (**bubbles\_on\_black\_background**):**

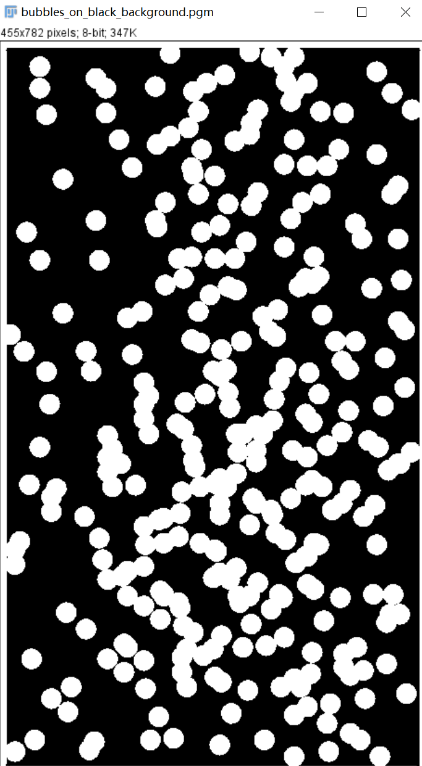
**Algorithm:**

It is given that all particles are of the same size. Determine the area (number of pixels) of a single particle; denote the area by A. Eliminate from the image the particles that were merged with the border of the image. Apply the connected component algorithm. Count the number of pixels in each component. A component is then designated as a single particle if the number of pixels is less than or equal to , where is a small quantity added to account for variations in size due to noise.

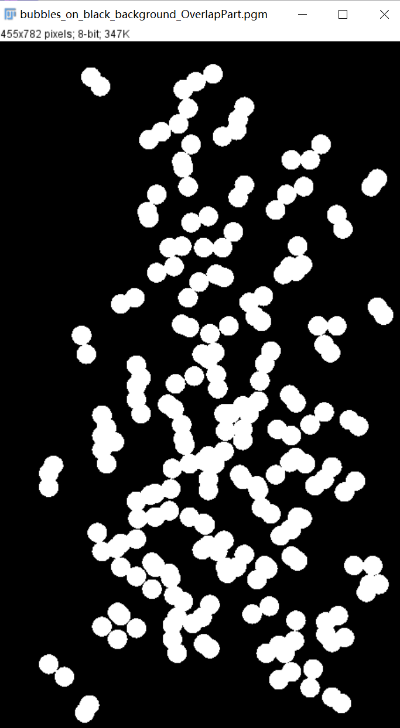
**Results (including pictures):**

Result of processing “bubbles\_on\_black\_background.pgm”:

Source Image:



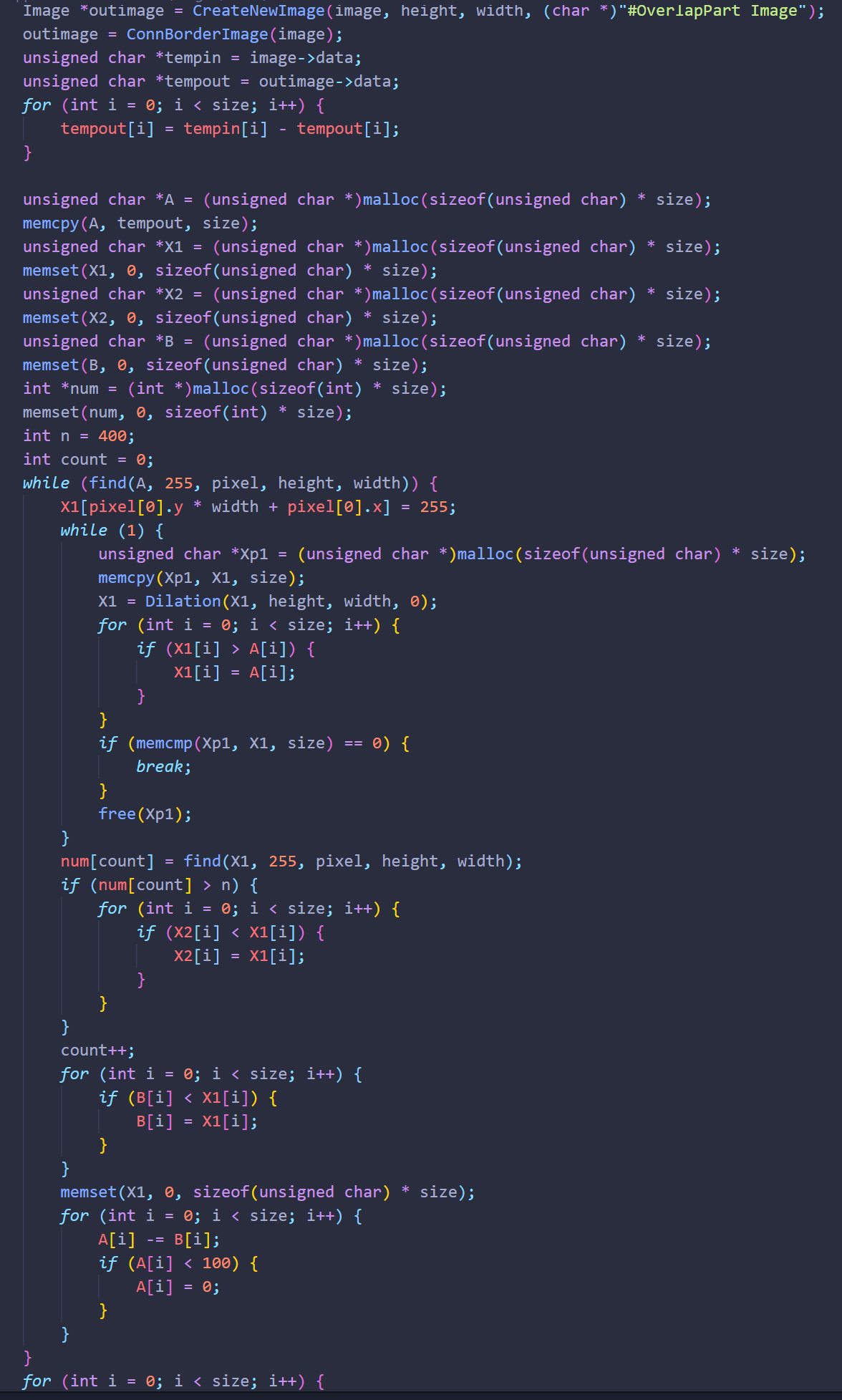
Result：



**Discussion:**

Extraction of connected components for a fully automated process

**Codes:**



1. **Only particles that not overlap each other (**bubbles\_on\_black\_background**):**

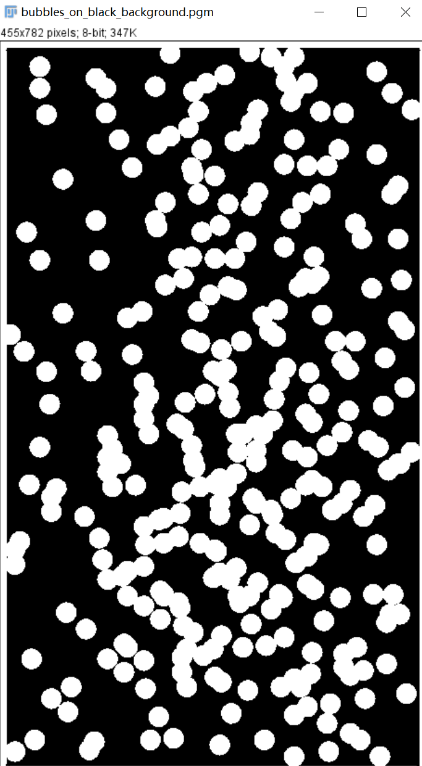
**Algorithm:**

Subtract from the image single particles and the particles that have merged with the border, and the remaining particles are overlapping particles.

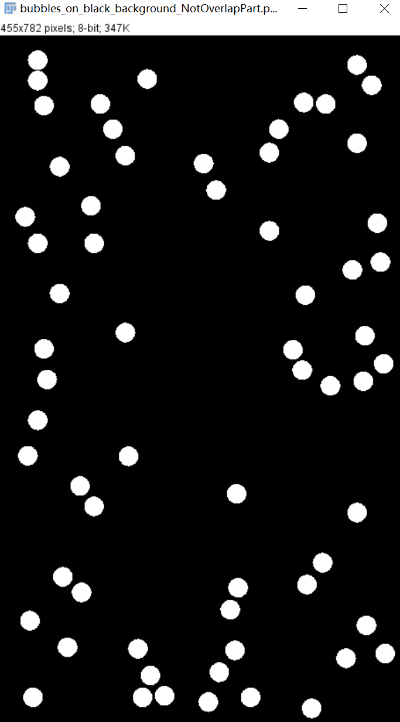
**Results (including pictures):**

Result of processing “bubbles\_on\_black\_background.pgm”:

Source Image:



Result：



**Discussion:**

When an operation is performed with this code, it cannot be performed at the same time as the previous one. The specific reason is that the *OverlapPartImage* function is called twice, resulting in some memory not being released when the C language manipulates the memory.

**Codes:**

