**ЛАБОРАТОРНА РОБОТА № 5**

**ОСНОВИ ЦИФРОВОГО ПРЕДСТАВЛЕННЯ ЗОБРАЖЕНЬ**

**Мета роботи** – ознайомитися з основними типами цифрових зображень, їхнім матричним представленням у середовищі **Scilab 2026.0.0** та набути практичних навичок роботи з базовими операціями обробки: читанням, збереженням, аналізом стиснення, перетворенням типів та аналізом пікселів.

## **5.1 Короткі теоретичні відомості**

### **5.1.1 Короткі відомості про пакеті Scilab**

#### 5.1.1.1 Система Scilab

Scilab – це система комп'ютерної математики, яка призначена для виконання інженерних і наукових обчислень, таких як:

* + - рішення нелінійних рівнянь і систем;
    - рішення задач лінійної алгебри;
    - рішення задач оптимізації;
    - диференціювання та інтегрування;
    - обробка експериментальних даних (інтерполяція і апроксимація, метод найменших квадратів);
    - рішення звичайних диференціальних рівнянь і систем.

Крім того, Scilab надає широкі можливості по створенню і редагуванню різних видів графіків і поверхонь. Незважаючи на те, що система Scilab містить достатню кількість вбудованих команд, операторів і функцій, відмінна її риса - це гнучкість.

Користувач може створити будь-яку нову команду або функцію, а потім використовувати її нарівні з вбудованими. До того ж, система має досить потужний власну мову програмування високого рівня, що говорить про можливість вирішення нових завдань.

Запуск Scilab можна здійснити по-різному в залежності від операційної системи. Але при стандартній установці ярлик програми завжди можна буде знайти в головному меню.

Робоче вікно Scilab складається з області меню і робочої області з командним рядком, що починається зі стрілки. Саме в командному рядку і відбувається рішення задачі. Робота в цьому режимі полягає в наступному.

Ви набираєте команду, для її виконання натискаєте клавішу Enter і отримуєте результат. Нижче Ви побачите чергове запрошення командного рядка: система готова виконувати наступний «наказ».

Наприклад, на рис. 1.1 показано виконання команди «4х8» тобто «Чотири помножити на вісім» і її виконання після натискання клавіші Enter «ans = 32.» - «відповідь 32. ». Набираєте чергову команду - Enter - отримуєте результат і так далі. Всі команди і результати, які знаходяться вище командного рядка, вже ніяк не змінити, вони доступні тільки для перегляду. Тому і область вище командного рядка називають областю перегляду.

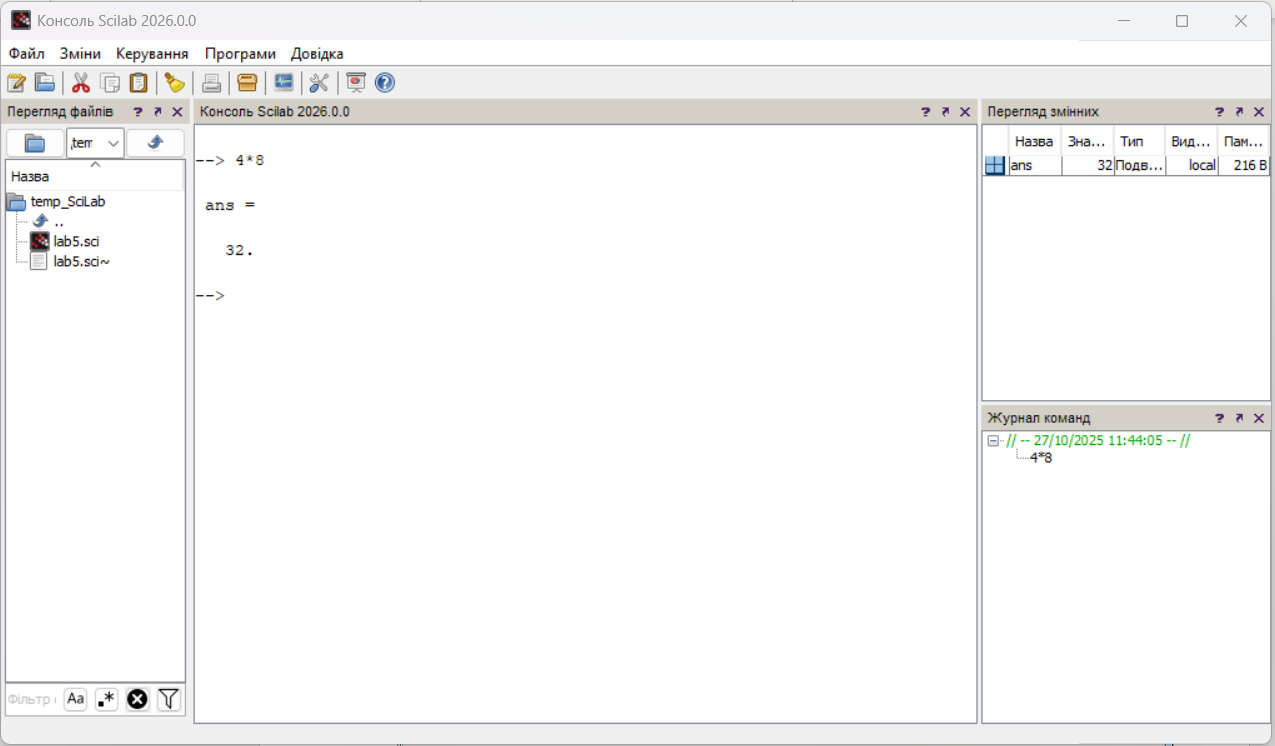


Рисунок 1.1 – Командне вікно програми Scilab.

Всі символи, що містяться в командному рядку, програма сприймає як текст команди або назва змінних, але якщо перед ними поставити подвійний слеш (//), то наступний за ним текст буде сприйнятий як коментар.

Доки натиснуто Enter, текст в командному рядку, текст можна редагувати стандартним способом. Після натискання Enter команда переміщається в область перегляду доступ до її редагування закритий. Якщо є необхідність повернутися до цієї чи іншої даної команді, то за допомогою клавіші управління курсором ↑ раніше виконані команди можуть бути повернуті в командний рядок. Після цього їх можна редагувати і виконувати повторно. При натисканні клавіші ↓ команди з'являються в командному рядку в зворотному порядку.

#### 5.1.1.2 Арифметичні обчислення

Для виконання найпростіших арифметичних операцій Scilab використовує такі оператори:

+ складання;

- віднімання;

\* множення;

/ ділення зліва направо;

\ ділення справа наліво;

^ піднесення до степеня.

Щоб обчислити значення арифметичного виразу, необхідно ввести його в командний рядок і натиснути Enter.

При записи вираження слід пам'ятати про пріоритет виконання арифметичних операцій. У разі необхідності його зміни використовуйте круглі дужки.

#### 5.1.1.3 Змінні

Користувач може визначити змінну, щоб використовувати її в подальших розрахунках. «Визначити змінну» означає поставити її ім'я і значення. З цією метою використовується оператор присвоювання, в загальному випадку має вигляд:

ім'я\_змінної = значення\_ змінної

У Scilab допускається використання в імені змінної безлічі символів, однак ми рекомендуємо використовувати тільки латинські букви і цифри, причому першою повинна бути буква. У цьому випадку Ви будете гарантовані від помилок, там більше що максимальна довжина імені - 24 символи. Система розрізняє великі і малі літери в іменах змінних, тобто ABC, abc, Abc- це імена різних змінних.

Вираз у правій частині оператора присвоювання може бути числом, арифметичним виразом, символьним виразом або рядком символів. В останньому випадку вираз справа береться в лапки.

#### 5.1.1.4 Системні змінні

Якщо користувач не поставив в командному рядку ім'я змінної, результат буде присвоєно системної змінної з ім'ям ans (від англ. *Answer* – відповідь). Змінну ans можна використовувати в подальших обчисленнях, але її значення буде змінюватися кожного разу після виконання команди без оператора присвоювання.

У Scilab існують інші системні змінні (правильніше - системні константи).

Їх запис починається символу%:

% і - уявна одиниця (√-1);

% рi - число π = 3.141592653589793;

% е - число e = 2.7182818;

% іnf - машинний символ нескінченності (∞);

% NaN - невизначений результат (0/0, ∞ / ∞ і т. П.);

% еps - умовний нуль% eps = 2.220Е-16.

#### 5.1.1.5 Елементарні математичні функції

Пакет Scilab забезпечений достатньою кількістю всіляких вбудованих функцій, знайомство з якими буде відбуватися в наступних розділах.

Тут наведемо лише елементарні математичні функції, які використовуються найчастіше (табл. 5.1).

Таблиця 5.1 – Елементарні математичні функції

|  |  |
| --- | --- |
| Функція | Опис функції |
| Тригонометричні | |
| sin(x)  cos(x)  tan(x)  cotg(x)  asin(x)  acos(x)  atan(x) | синус числа x  косинус числа x  тангенс числа x  котангенс числа x  арксинус числа x  арккосинус числа x  арктангенс числа x |
| Експоненціальні | |
| exp(x)  log(x) | Експонента числа x  Натуральний логарифм числа x |
| Інші | |
| sqrt(x)  abs(x)  log10(x)  log2(x) | корінь квадратний з числа x модуль числа x  десятковий логарифм від числа x  логарифм за основою два від числа x |

### **5.1.2 Фізична природа та математична модель цифрового зображення**

*Цифрове зображення* – це результат перетворення неперервного оптичного сигналу у дискретну форму, придатну для зберігання та обробки комп'ютером. Цей процес моделюється як двовимірна дискретна функція *f*(*x*, *y*), де:

* *x* та *y* **–** просторові координати (стовпці та рядки), які набувають цілочисельних значень.
* *f*(*x*, *y*) **–** значення функції в цій точці, яке називається *інтенсивністю***,** *яскравістю* або *рівнем сірого*.

Перехід від аналогового світу до цифрового зображення здійснюється двома ключовими процесами:

1. *дискретизація простору* **–** встановлення сітки пікселів. Це визначає просторову роздільну здатність (*M* х *N*).
2. *квантування інтенсивності* **–** призначення кінцевого числа значень (наприклад, 28 = 256 рівнів) для опису яскравості/кольору кожного пікселя.

### **5.1.3 Типи зображень як багатовимірні матриці у Scilab**

Фундаментальним для цифрової обробки є те, що у програмних середовищах (таких як Scilab) зображення інтерпретується як *числовий багатовимірний масив* (*матриця*)*.* Усі операції з обробки зводяться до маніпуляцій елементами цього масиву.

Таблиця 5.2 – Інтерпретаія зображення в Scilab

|  |  |  |  |
| --- | --- | --- | --- |
| Тип зображення | Опис | Матрична структура у Scilab | Глибина та діапазон |
| Повнокольорове (RGB) | Зображення складається з трьох незалежних шарів (каналів): Червоного (R), Зеленого (G) та Синього (B). Сукупність їхніх інтенсивностей формує колір пікселя. | Тривимірний масив (M х N х3). M **–** рядки, N **–** стовпці, 3 **–** канали. | 24 біти (по 8 біт на канал), діапазон $0...255$. |
| Півтонове (Grayscale) | Зображення має лише один канал, де значення пікселя описує інтенсивність сірого: 0 відповідає чорному кольору, 255 — білому. | Двовимірна матриця (M х N х 1). | 8 біт, діапазон $0...255$. |

### **5.1.4 Приклади базових операцій Scilab**

* 1. *Введення-виведення зображень*

Для початку роботи необхідно завантажити файл у пам'ять, перетворивши його на масив (табл. 5.3).

Таблиця 5.3 – Команди введення виведення зображень

|  |  |  |
| --- | --- | --- |
| Дія | Команда Scilab | Пояснення |
| ЧИТАННЯ | f = imread('image.jpg'); | Вбудована функція завантажує файл у змінну f. |
| ПЕРЕВІРКА | size(f) | Якщо зображення кольорове, результатом буде, наприклад, [600, 800, 3]. |
| ВІДОБРАЖЕННЯ | imshow(f); | Відображає вміст матриці f у графічному вікні. |

* 1. *Перетворення кольорового зображення у півтонове*

Це є ключовим етапом для переходу до обробки, оскільки більшість подальших алгоритмів (як, наприклад, розрахунок яскравості) працюють з одноканальними зображеннями. Перетворення являє собою редукцію тривимірного масиву до двовимірного за формулою зваженого усереднення:



У Scilab 2026.0.0 це здійснюється за допомогою однієї потужної вбудованої функції **grayImage = im2gray(f).** Вона виконує зважене перетворення кольорової матриці f у півтонову матрицю grayImage.

* 1. *Обробка та маніпуляції з матрицею (індексація)*

Після отримання півтонової матриці grayImage усі подальші дії **–** це маніпуляції її елементами:

1. *обчислення загальної яскравості* **–** для визначення загальної світлості або темноти зображення обчислюється середнє значення усіх пікселів.

// grayImage(:) розгортає матрицю в один вектор

averageBrightness = mean(double(grayImage(:)));

1. *локальна модифікація (нанесення мітки)* **–** доступ до регіону зображення здійснюється через індексацію. Оператор «**:»** (двокрапка) використовується для виділення діапазону рядків та стовпців.

// Присвоєння значення 255 (білий) квадрату 20x20

// у верхньому лівому куті

grayImage(1:20, 1:20) = 255;

## **5.2 Практична робота**

1. *Підготовка середовища та введення зображення*

Перш ніж розпочати обробку, необхідно ініціалізувати робочу директорію та завантажити зображення, перетворивши його у числовий масив

// Встановлюємо ім'я файлу та директорію для

// результатів.

// Створюємо необхідні підпапки, щоб уникнути помилок.

filename = 'Portrait.jpg';

output\_dir = 'C:\Lab5\_Results';

if ~isdir(output\_dir) then

mkdir(output\_dir);

end

scf(1);

// Функція imread зчитує файл і зберігає

//його у змінну f як тривимірний масив (RGB).

// Функція imshow(f) відображає масив f   
//як зображення у графічному вікні.

imshow(f);

title('1. Початкове кольорове зображення');

1. *Аналіз стиснення (JPG vs. PNG)*

На цьому етапі ми порівняємо стиснення зображень з втратами (JPEG) та без втрат (PNG), використовуючи інформацію про розміри файлів.

Зберігаємо зображення у двох форматах. imwrite приймає масив і зберігає його у файл:

imwrite(f, fullfile(output\_dir, 'Photo\_RGB.jpg'), 'jpg');

imwrite(f, fullfile(output\_dir, 'Photo\_RGB.png'), 'png');

Функція imfinfo отримує метадані файлу, зокрема розмір (FileSize), ширину (Width), висоту (Height) та розрядність (BitDepth).

K1 = imfinfo(fullfile(output\_dir, 'Photo\_RGB.jpg'));

K2 = imfinfo(fullfile(output\_dir, 'Photo\_RGB.png'));

Обчислюємо міру стиснення:

imbyte = K1.Width \* K1.Height \* K1.BitDepth / 8;

KS\_1 = imbyte / K1.FileSize;

KS\_2 = imbyte / K2.FileSize;

KS показує, у скільки разів стиснений файл менший за його теоретичний нестиснений розмір. JPEG (стиснення з втратами) завжди матиме значно більший KS порівняно з PNG (стиснення без втрат) для фотографічних зображень, оскільки JPEG відкидає візуально надлишкову інформацію.

1. *Перетворення типів зображень (Grayscale, Binary*)

Оскільки більшість алгоритмів обробки працюють з одним каналом, необхідно перетворити RGB-зображення у півтонове.

Функція im2gray виконує зважене усереднення трьох каналів (R, G, B) за формулою Y = 0.299R + 0.587G + 0.114B, редукуючи тривимірний масив до двовимірної матриці.

G = im2gray(f); scf(2);

imshow(G);

title('2. Півтонове зображення (Grayscale)');

Функція im2bw (Image to Binary) встановлює пікселю значення 1 (білий), якщо його яскравість вища за заданий поріг, і 0 (чорний), якщо нижча. Пороги задаються у діапазоні [0, 1].

Gb\_50 = im2bw(G, 0.50); scf(3); imshow(Gb\_50); title('3. Двійкове зображення (Поріг 50%)');

Перетворення в *Grayscale* дозволяє аналізувати зображення за одним параметром (яскравістю), а перетворення в Binary використовується для найпростішої форми сегментації, де об'єкти відділяються від фону за рівнем освітленості.

1. *Розбиття на бітові площини*

Кожен 8-бітний піксель сірого зображення є числом від 0 до 255. Це число складається з 8 бітів. Розбиття на бітові площини дозволяє побачити внесок кожного біта у фінальне зображення.

Функція bitget(C, i) повертає значення i-го біта кожного пікселя матриці C.

C = im2uint8(G); B8 = bitget(C, 8);

Біти 1-4 (нижчі) містять дрібний шум, тоді як біти 5-8 (вищі) містять основну візуальну інформацію (контури та форми).

scf(4); imshow(logical(B8)); title('4. Бітова площина №8 (Старший біт)');

Бітові площини демонструють ієрархію інформації: старші біти (7-й, 8-й) відповідають за глобальну структуру зображення, а молодші біти (1-й, 2-й) відповідають переважно за дрібні деталі та шум.

1. *Аналіз сусідів пікселів*

Операції обробки зображень (наприклад, фільтрація, згортка) рідко працюють з окремим пікселем; вони використовують значення його сусідів. Цей крок демонструє прямий доступ до матричних елементів.

Четвірка (N4) – сусіди зверху, знизу, зліва, справа:

p1 = [21, 17]; N1 = [grayImage(p1(1) - 1, p1(2)); ...]

Вісімка (N8) – всі 8 пікселів навколо заданого центрального пікселя:

p3 = [19, 88]; N3 = [grayImage(p3(1) - 1, p3(2) - 1); ...]

Отримані значення N1, N2, N3 є локальним вікном даних, яке є вхідним для більшості фільтрів та алгоритмів. Операції з зображеннями – це послідовне застосування такої операції до кожного пікселя.

## **5.3 Індивідуальне завдання**

Спочатку Ви виконуєте загальну практичну частину, а потім реалізуєте наступні кроки, виходячи зі свого номера варіанта за журналом.

1. *Визначення координат та рівня яскравості*

Спочатку необхідно обчислити середній рівень яскравості, який є критерієм для нанесення міток.

Розрахунок яскравості:

averageBrightness = mean(double(grayImage(:)));

Визначення кольору мітки:

if averageBrightness < 128 then markColor = 255; else markColor = 0; end

1. *Завдання за варіантами*

Виконайте наступні пункти відповідно до вашого номера варіанта.

Таблиця 5.1 – **ЗАВДАННЯ 1.** Вирізання та аналіз сусідів

|  |  |  |
| --- | --- | --- |
| Варіант | Розмір вирізаної області (центр зображення) | Координати пікселя для аналізу N8 (Вісімка сусідів) |
| 1 | 100 | p(50, 50) |
| 2 | 120 | p(60, 60) |
| 3 | 80 | p(70, 70) |
| 4 | 150 | p(80, 80) |
| 5 | 90 | p(90, 90) |
| 6 | 110 | p(100, 100) |
| 7 | 130 | p(110, 110) |
| 8 | 70 | p(120, 120) |
| 9 | 140 | p(130, 130) |
| 10 | 60 | p(140, 140) |
| 11 | 100 | p(50, 50) |
| 12 | 120 | p(60, 60) |
| 13 | 80 | p(70, 70) |
| 14 | 150 | p(80, 80) |
| 15 | 90 | p(90, 90) |
| 16 | 110 | p(100, 100) |
| 17 | 130 | p(110, 110) |
| 18 | 70 | p(120, 120) |
| 19 | 140 | p(130, 130) |
| 20 | 60 | p(140, 140) |

Код для вирізання (приклад для 100x100):

[rows, cols] = size(grayImage);

crop\_size = 100; // ЗАМІНИТИ на розмір варіанта

half\_crop = floor(crop\_size / 2);

startRow = round(rows / 2) - half\_crop + 1;

startCol = round(cols / 2) - half\_crop + 1;

croppedImage = grayImage(startRow:startRow + crop\_size - 1, startCol:startCol + crop\_size - 1);

scf(12); imshow(croppedImage); title(['Центральна область ' + string(crop\_size) + 'x' + string(crop\_size)]);

imwrite(croppedImage, fullfile(output\_dir, 'Crop', 'Photo\_cropped\_VAR\_' + string(crop\_size) + '.jpg'), 'jpg');

Таблиця 5.2 – **ЗАВДАННЯ** **2.** Квантування та Дискретизація

|  |  |  |
| --- | --- | --- |
| Варіант | Кількість рівнів квантування (L) | Ядро дискретизації (блочне усереднення K×K) |
| 1, 11 | 16 | 10 |
| 2, 12 | 32 | 20 |
| 3, 13 | 64 | 5 |
| 4, 14 | 4 | 50 |
| 5, 15 | 128 | 15 |
| 6, 16 | 16 | 25 |
| 7, 17 | 32 | 30 |
| 8, 18 | 64 | 40 |
| 9, 19 | 4 | 12 |
| 10, 20 | 128 | 35 |

Код для квантування (для L=16):

levels = 16; // ЗАМІНИТИ на L варіанта

quantizedImage = floor(double(G) / (256 / levels)) \* (256 / levels);

scf(13); imshow(uint8(quantizedImage)); title(['Квантування на ' + string(levels) + ' рівнів']);

Для дискретизації використовується блочне усереднення. Оскільки blkproc відсутній, використовується функція, яка зменшує зображення, а потім повертає його до початкового розміру методом найближчого сусіда, створюючи ефект "пікселізації". Код для дискретизації:

block\_size = 10; // ЗАМІНИТИ на К варіанта

// Зменшення: обчислюємо середнє значення блоків

G\_small = imresize(G, size(G)/block\_size, 'average');

// Збільшення: повертаємо до оригінального розміру, використовуючи значення блоку

G\_discretized = imresize(G\_small, size(G), 'nearest');

scf(14); imshow(G\_discretized, []); title(['Дискретизація ' + string(block\_size) + 'x' + string(block\_size)]);

**ЗАВДАННЯ 3.** Нанесення міток

Нанесіть квадратні мітки розміром 20 x 20 пікселів:

1. у центр зображення;
2. по кутах зображення.

Логіка кольору мітки:

* Якщо середня яскравість averageBrightness < 128 → Колір мітки БІЛИЙ (255).
* Якщо середня яскравість averageBrightness > 128 → Колір мітки ЧОРНИЙ (0).

Код для нанесення міток (використання markColor):

// Використовувати markColor, визначений у п. II.1

markSize = 20;

[rows, cols] = size(grayImage);

centerRow = round(rows / 2);

centerCol = round(cols / 2);

// Нанесення на центр та кути

grayImage(max(1, centerRow - 9):min(rows, centerRow + 10), max(1, centerCol - 9):min(cols, centerCol + 10)) = markColor;

grayImage(1:markSize, 1:markSize) = markColor;

grayImage(1:markSize, cols-markSize+1:cols) = markColor;

grayImage(rows-markSize+1:rows, 1:markSize) = markColor;

grayImage(rows-markSize+1:rows, cols-markSize+1:cols) = markColor;

scf(15); imshow(grayImage); title('7. Зображення з фінальними мітками');

imwrite(grayImage, fullfile(output\_dir, 'Marks', 'Photo\_with\_marks\_VAR\_' + string(V) + '.jpg'), 'jpg');

## **5.4 Зміст звіту**

* Титульний аркуш
* Тема та мета роботи
* Код виконаного практичного завдання із скриншотами отриманих результатів.
* Код виконаного індивідуального завдання із скриншотами отриманих результатів.
* Висновки.
* Відповіді на контрольні питання.

## **5.5 Контрольні питання**

1. Назвіть основні напрямки інженерних та наукових обчислень, для яких призначена система Scilab? Яка її головна відмінна риса (гнучкість, що дозволяє створювати нові функції)?
2. Що таке цифрове зображення з точки зору комп'ютерної графіки? Який основний елемент формує зображення на екрані чи в матриці?
3. Як чорно-біле (півтонове) зображення представляється у пам'яті комп'ютера? Що означає значення елемента матриці (пікселя) у цьому випадку?
4. Поясніть принцип представлення кольорового зображення в моделі RGB. Скільки матриць (каналів) використовується для зберігання кольору і за що відповідає кожна з них?
5. Що таке квантування зображення? Як змінюється глибина кольору (наприклад, з 256 рівнів на 16) і на що це впливає?
6. Поясніть сутність процесу дискретизації у контексті обробки зображень. Як параметр block\_size (як у Завданні 2) впливає на візуальний ефект "пікселізації"?
7. Що таке індексоване кольорове зображення? Як воно допомагає зменшити обсяг пам'яті, необхідний для зберігання файлу?
8. Чому у більшості операцій обробки зображень використовується діапазон значень пікселя від 0 до 255? Що відповідає значенню 0, а що — 255?
9. Опишіть логіку, за якою у Завданні 3 визначається колір мітки? Від якого критерію залежить, чи буде мітка чорною (0), чи білою (255)?
10. Назвіть та опишіть призначення двох-трьох основних функцій Scilab, необхідних для читання, відображення та аналізу цифрового зображення (наприклад, imread, imshow, size).