МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Український державний університет імені Михайла Драгоманова

Факультет математики, інформатики та фізики

Кафедра інформаційних технологій та програмування

**Звіт**

з лабораторної роботи №9

«Класичні алгоритми»

з дисципліни «Програмування»

Виконала:

студентка ІІІ курсу групи 31І

Мотовилець М. Т.

Перевірила:

викладач Устименко О.Б.

Оцінка\_\_\_\_\_\_\_\_\_\_\_

Дата \_\_\_\_\_\_\_\_\_\_\_\_\_

Київ - 2024

Зміст

[**Мета роботи** 4](#_Toc168168315) [4](#_Toc168168316)

[**Постановка задачі** 5](#_Toc168168317)

[**Основна частина** 6](#_Toc168168320)

[**2.1 Опис вхідних та вихідних даних** 6](#_Toc168168321)

[. 7](#_Toc168168336)

[**2.2 Блок-схема** 7](#_Toc168168337)

[**Висновки** 8](#_Toc168168338)

[8](#_Toc168168339)

[**Додатки(лістинги програми)** 9](#_Toc168168340)

[9](#_Toc168168341)

**Мета роботи**

Мета лабораторної роботи полягає у вивченні та практичному застосуванні класичних алгоритмів. Під час виконання роботи ознайомитись з основними концепціями та методами, що лежать в основі класичних алгоритмів.

**Постановка задачі**

Варіант №14

Напишіть програму, яка реалізує класичний алгоритм тернарного пошуку заданого елементу у стовпцях двовимірного масиву. Розмірність масиву та всі елементи генеруються за допомогою випадкових чисел.

**Основна частина**

**2.1 Опис вхідних та вихідних даних**

**Вхідні дані**

Розміри матриці:

* rows (кількість рядків): встановлюється на 5.
* cols (кількість стовпців): встановлюється на 4.

Значення елементів матриці:

* Генеруються випадкові цілі числа в діапазоні від 1 до 100 для кожного елемента матриці.

Значення для пошуку:

* target (цільове значення для пошуку): користувач вводить це значення вручну через команду input().

**Вихідні дані:**

Відсортована матриця:

* Відображається на екрані після сортування кожного стовпця.

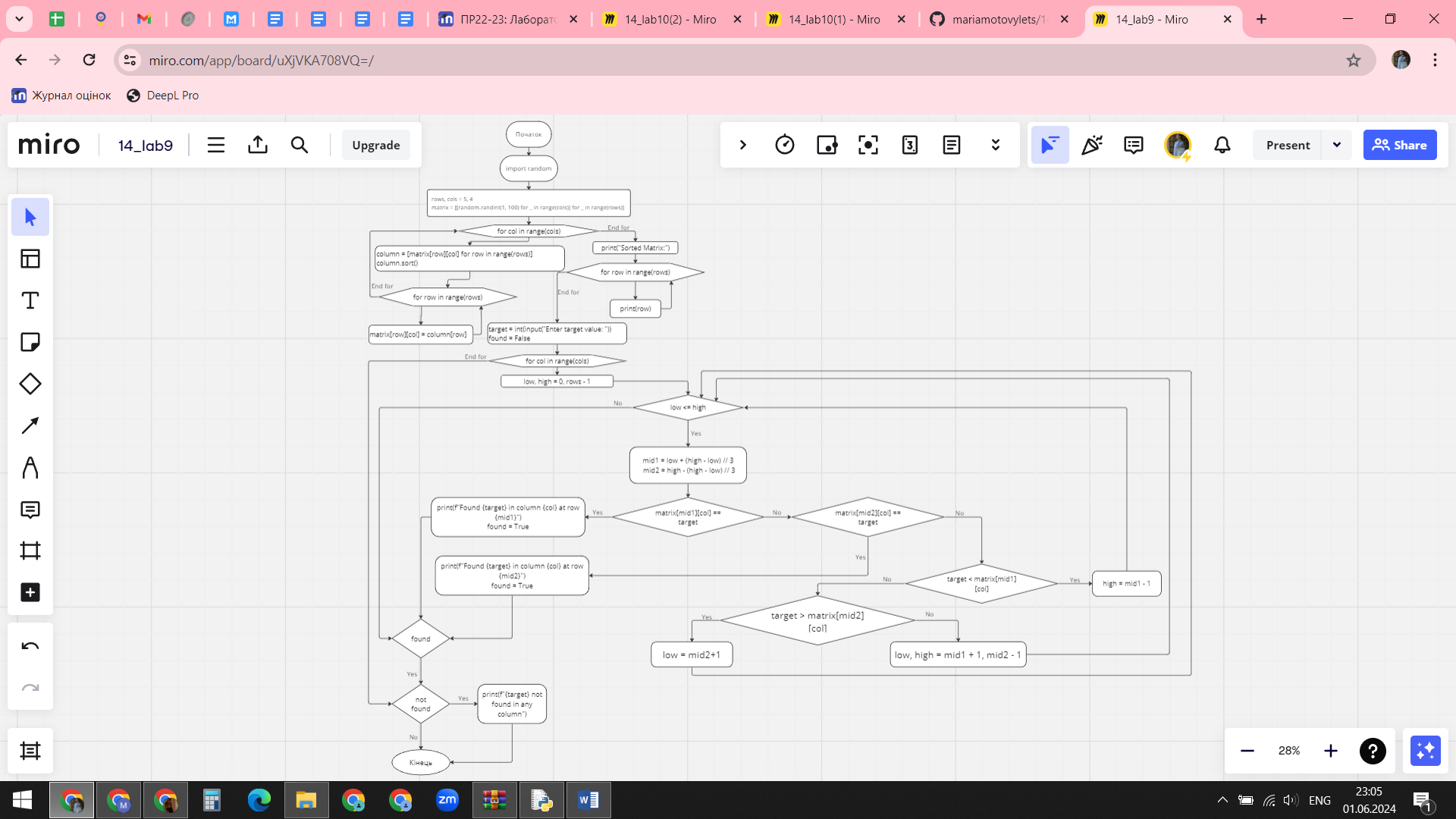
Результат пошуку:

* Якщо цільове значення знайдено: повідомлення у форматі: Found {target} in column {col} at row {mid1} або Found {target} in column {col} at row {mid2}, де target - значення, яке шукається, col - індекс стовпця, а mid1 або mid2 - індекс рядка, де значення було знайдено.
* Якщо цільове значення не знайдено: повідомлення: {target} not found in any column.

.

**2.2 Блок-схема**

<https://miro.com/welcomeonboard/RXZzNVhrOWdhSFowWlBvTXJLUDBWNFlZaUpmRmZpZU9FSzh6UGMwRUhHaWpPbTRmY0FBWnE4ZVM3cGRmU1EwUXwzNDU4NzY0NTQ1NDc4OTEzMjUyfDI=?share_link_id=83841220178>

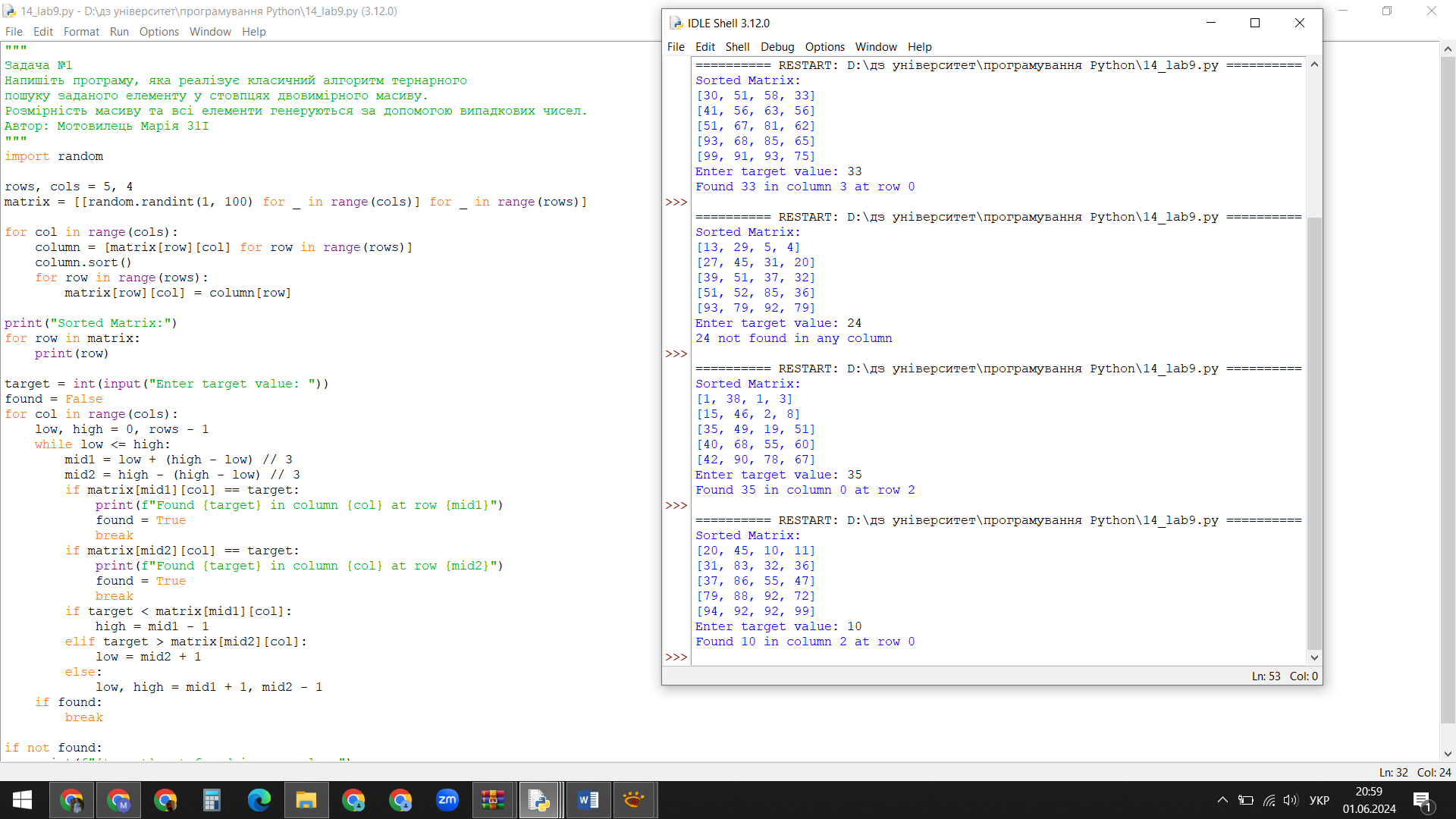


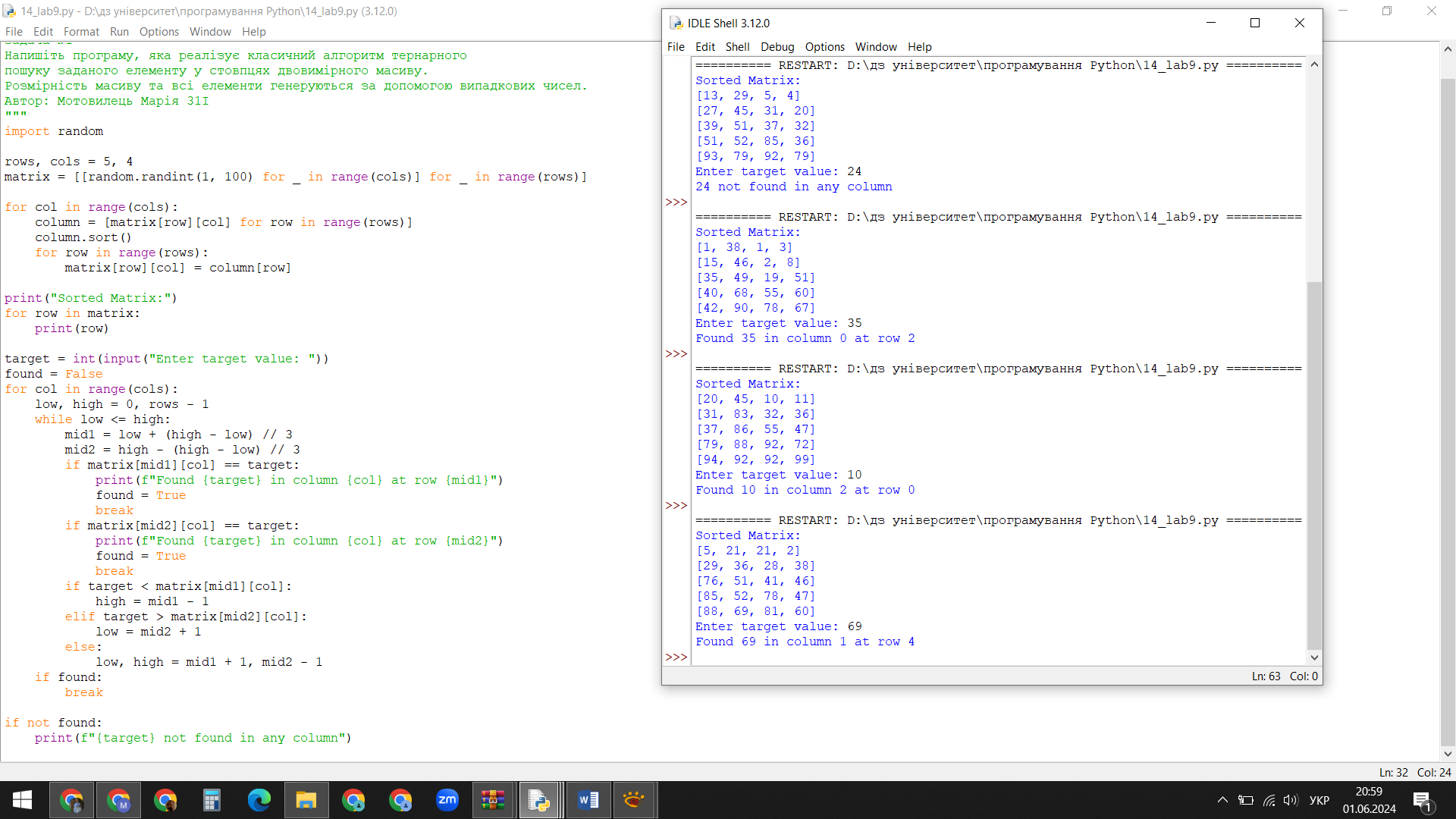
**Висновки**

В результаті виконання лабораторної роботи було здобуто глибше розуміння класичних алгоритмів і їх застосування у різних областях обчислювальної науки. Ознайомлення з такими основними алгоритмами, як сортування та робота з двовимірними масивами, дозволило навчитися їх реалізовувати в коді. Аналіз ефективності алгоритмів допоміг усвідомити важливість вибору правильного алгоритму для конкретного завдання з огляду на його часову та просторову складність. Лабораторна робота також сприяла розвитку алгоритмічних навичок і здатності до абстрактного мислення, що є важливими аспектами комп'ютерної науки та інформаційних технологій.

**Додатки(лістинги програми)**

Посилання на репозиторій: <https://github.com/mariamotovylets/14_lab9_Motovylets>





import random

rows, cols = 5, 4

matrix = [[random.randint(1, 100) for \_ in range(cols)] for \_ in range(rows)]

for col in range(cols):

column = [matrix[row][col] for row in range(rows)]

column.sort()

for row in range(rows):

matrix[row][col] = column[row]

print("Sorted Matrix:")

for row in matrix:

print(row)

target = int(input("Enter target value: "))

found = False

for col in range(cols):

low, high = 0, rows - 1

while low <= high:

mid1 = low + (high - low) // 3

mid2 = high - (high - low) // 3

if matrix[mid1][col] == target:

print(f"Found {target} in column {col} at row {mid1}")

found = True

break

if matrix[mid2][col] == target:

print(f"Found {target} in column {col} at row {mid2}")

found = True

break

if target < matrix[mid1][col]:

high = mid1 - 1

elif target > matrix[mid2][col]:

low = mid2 + 1

else:

low, high = mid1 + 1, mid2 - 1

if found:

break

if not found:

print(f"{target} not found in any column")