

Міністерство освіти і науки України
Донецький національний університет імені Василя Стуса Факультет
інформаційних і прикладних технологій
Кафедра інформаційних технологій

ЗВІТ

з лабораторної роботи № 9
з дисципліни «Основи програмування»
на тему:
«Обробка одновимірних масивів»

Виконав: студент гр. Б25_д/F3

Кручківський Ю.О.

Перевірив: доц. Бабаков Р. М.

Варіант 7:

Завдання	
1	серед елементів, що лежать у діапазоні [0; 100], кількість оцінок «А», кількість оцінок «В», кількість оцінок «С», кількість оцінок «D», кількість оцінок «Е», кількість оцінок «FX»
2	видалити з масиву усі елементи, рівні максимальному або мінімальному елементам масиву

Завдання 1:

Код до завдання 1:

```
from pprint import pprint
from abc import ABC, abstractmethod
from enum import Enum, unique, auto

class TypeECTS(Enum): # Визначаємо клас тип даних для оцінок за
таблицею ECTS
    A = auto()
    B = auto()
    C = auto()
    D = auto()
    E = auto()
    FX = auto()

class SchemeECTS: # Визначаємо клас контейнер для збереження схеми
оцінювання
    def __init__(self, type_, **data):
        self.type = type_
        self.data = data

    def __repr__(self):
        return f"{self.type}: {self.data}"

@unique
class RatingECTS(Enum): # Визначаємо клас що реєструє схеми оцінювання
    A = SchemeECTS(TypeECTS.A, threshold=90)
    B = SchemeECTS(TypeECTS.B, threshold=82)
    C = SchemeECTS(TypeECTS.C, threshold=75)
    D = SchemeECTS(TypeECTS.D, threshold=67)
    E = SchemeECTS(TypeECTS.E, threshold=60)
    FX = SchemeECTS(TypeECTS.FX, threshold=0)

class IRating(ABC): # Визначаємо інтерфейс оцінки
    @property
    @abstractmethod
    def ECTS(self) -> SchemeECTS: # визначаємо абстрактний метод для
оцінки за таблицею ECTS
        pass

class Rating(IRating): # ініціалізуємо клас модель оцінки
    def __init__(self, numeric_rating: int): # реалізовуємо метод
конструктор для класу
        self.numeric = numeric_rating
        self._ects = self.__calc_ECTS_rating(self.numeric) # задаємо
оцінку за таблицею ECTS
```

```

@property
def ECTS(self) -> RatingECTS:
    return self._ects

    @staticmethod
    def __calc_ECTS_rating(numeric_rating: int): # оголошуємо статичний
метод для обрахунку оцінки за таблицею ECTS
        for ects in RatingECTS: # йдемо по списку оцінки за ECTS та її
порозу
            if numeric_rating >= ects.value.data["threshold"]: # якщо
задана оцінка >= порогу повертаємо її букву за таблицею ECTS
                return ects
            raise ValueError("Rating out of range") # якщо оцінка залишилась
не обробленою викидаємо помилку про її не відповідність заданим правилам

    def __repr__(self): # реалізуємо метод що визначає строкове
визначення для об'єкту оцінки
        return "rating: " + self.ECTS.name + " - " + str(self.numeric)

class RatingCounter: # ініціалізуємо клас лічильник
    def __init__(self): # реалізуємо метод конструктор для класу
        self.table = {} # ініціалізуємо хеш-таблицю для лічильника

    def __write_rating(self, rating: IRating): # оголошуємо
інкапсульований метод що визначає логіку при додаванні оцінки до
лічильника
        self.table[rating.ECTS] = self.table.get(rating.ECTS, 0) + 1 #
намагаємось отримати значення за ключем, якщо запис існує виконуємо +1
якщо ні то створюємо запис та встановлюємо значення на 0

    def add(self, rating: IRating): # оголошуємо метод що визначає
поведінку при додаванні оцінки до лічильника
        self.__write_rating(rating)

    def get_pretty_table(self): # оголошуємо метод що повертає строкове
визначення таблиці для об'єкту лічильника
        return "\n".join([f"{ects.name}\t: {score}" for ects, score in
list(self.table.items())])

    def __repr__(self): # реалізуємо метод що визначає строкове
визначення для об'єкту лічильника
        return pformat(self.table)

ratings = list(range(1, 101)) # ініціалізуємо заданий діапазон оцінок

if not ratings: # перевірка на валідність діапазону
    raise ValueError("Ratings list not initialized") # в разі хибного
діапазону викидаємо помилку

counter = RatingCounter() # ініціалізуємо клас лічильник
for numeric_rating in ratings: # йдемо по елементах діапазону
    counter.add( # додаємо оцінку до лічильника
        Rating(numeric_rating) # ініціалізуємо оцінку з заданого числа
діапазону
    )

```

```
print(f"Таблиця підрахунку оцінок в діапазоні [{ratings[0]}; {ratings[-1]}]")
print(counter.get_pretty_table())
```

Блок схема до завдання 1:

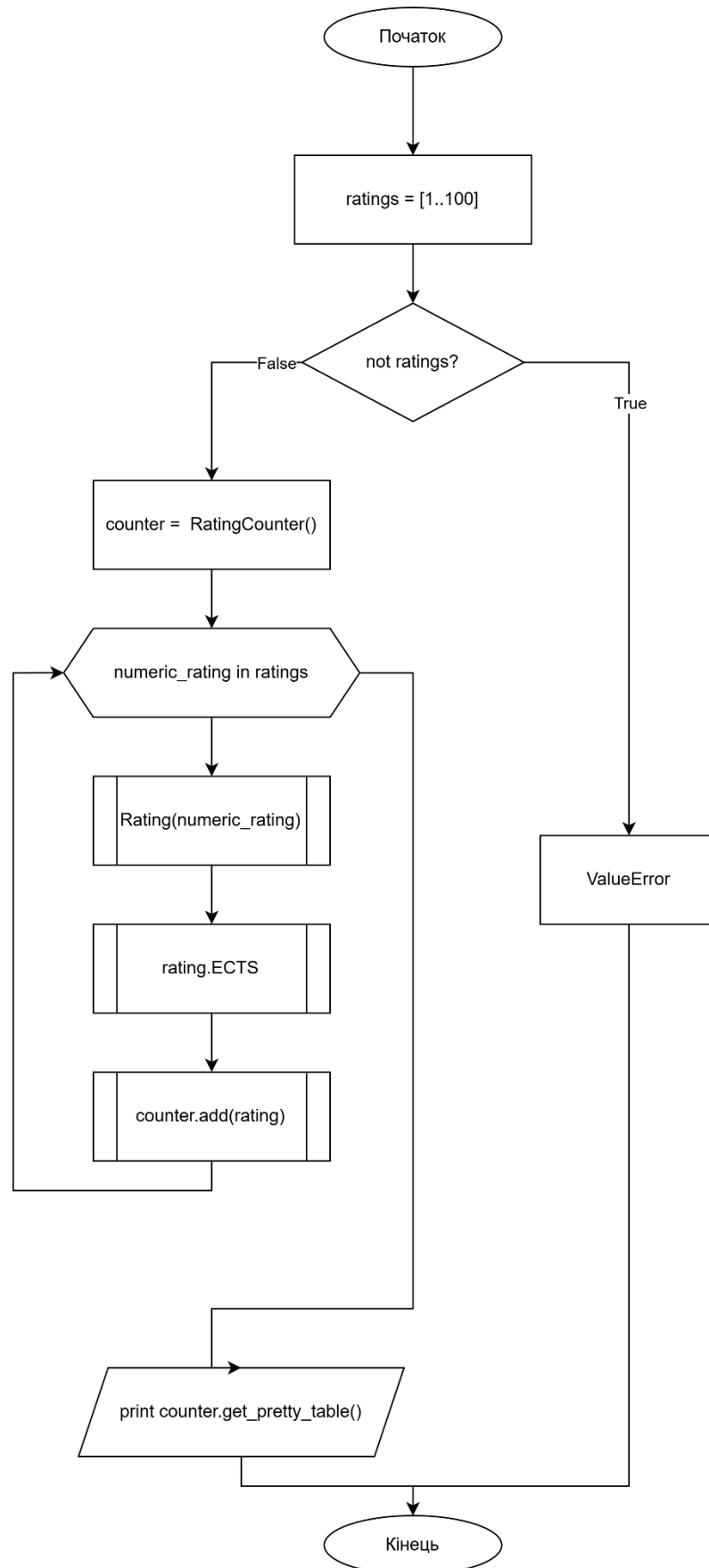


Рисунок 1 - Блок-схема логіки завдання 1



Рисунок 2 - Блок-схема обрахунку ects

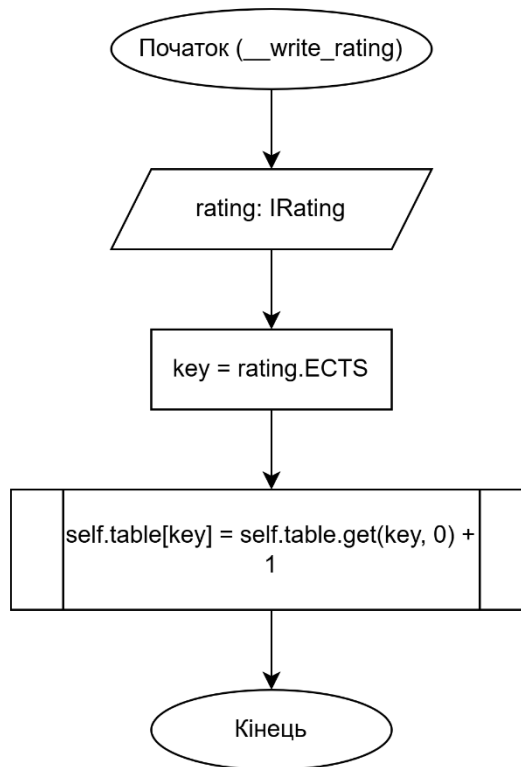


Рисунок 3 - Блок-схема запису в таблицю до завдання 1

Тестування завдання 1:

Таблиця підрахунку оцінок в діапазоні [1; 100]

FX	: 59
E	: 7
D	: 8
C	: 7
B	: 8
A	: 11

Завдання 2:

Код до завдання 2:

```
import random

def array_gen(length=10): # Для прикладу визначаємо ф-цію для генерації
    випадкових масивів заданої довжини
    return [random.randint(0, 100) for _ in range(length)]

def find_min(array: list): # Визначає ф-цію пошуку мінімального елементу
    масиву
    min_i = array[0] # Вважаємо перший елемент мінімальний
    for i in array: # Йдемо по елементах списку
        if i < min_i:
            min_i = i # Якщо елемент менший за минулий мінімальний
    вважаємо його мінімальним
    return min_i

def find_max(array: list): # Визначає ф-цію пошуку максимального
    елементу масиву
    max_i = array[0] # Вважаємо перший елемент максимальний
    for i in array:
        if i > max_i:
            max_i = i # Якщо елемент менший за минулий максимальний
    вважаємо його мінімальним
    return max_i

example_array = array_gen()

if not example_array:
    raise ValueError("Array is not initialized") # Викидаємо по милку у
    разі не валідного масиву

max_of_array = find_max(example_array)
min_of_array = find_min(example_array)

print(f"Початковий масив: {example_array}")
print(f"Мінімальний елемент: {min_of_array}")
print(f"Максимальний елемент: {max_of_array}")

example_array.remove(max_of_array)
example_array.remove(min_of_array)

print(f"Початковий масив після видалення: {example_array}")
```

Блок-схема до завдання 2:

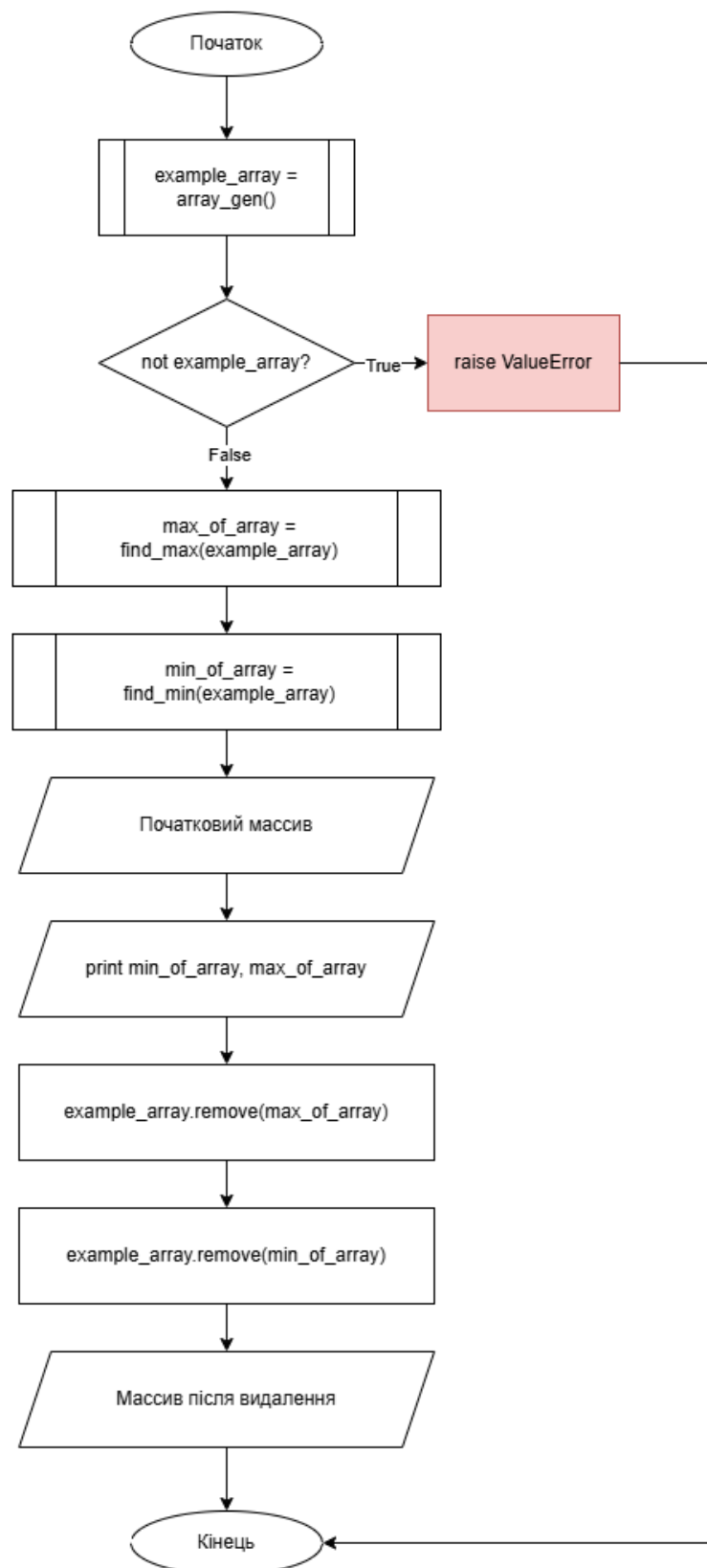


Рисунок 4 - Блок-схема логіки завдання 2

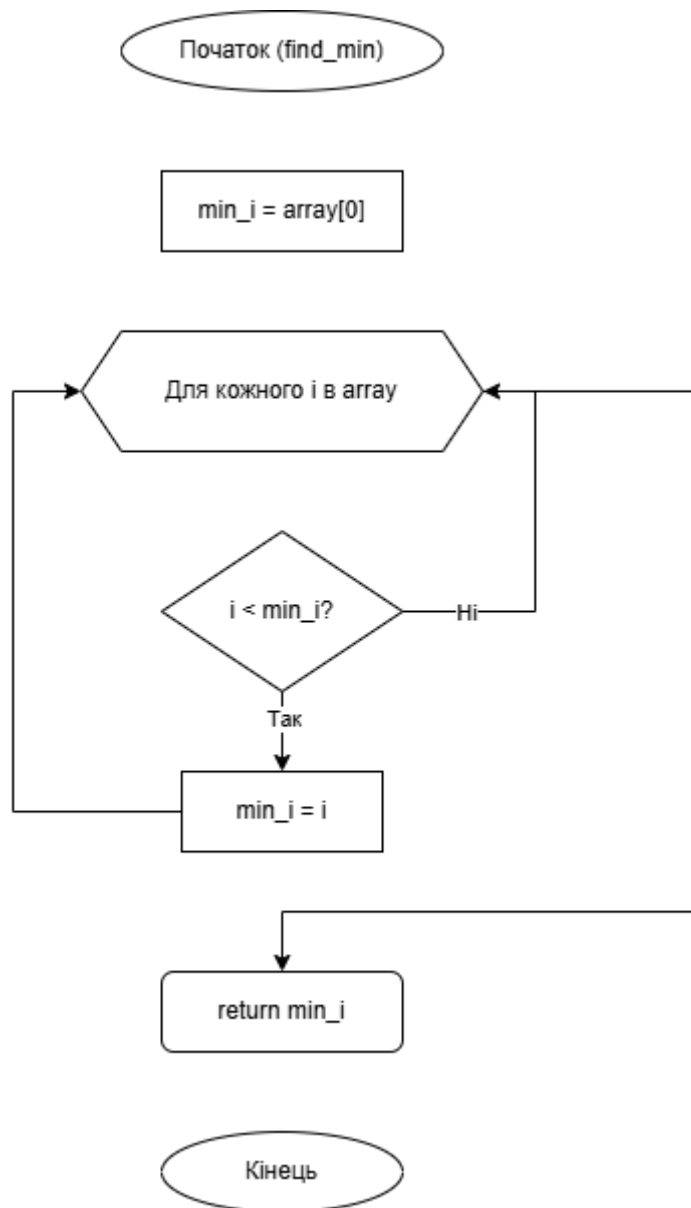


Рисунок 5 - Блок-схема знаходження мінімального до завдання 2

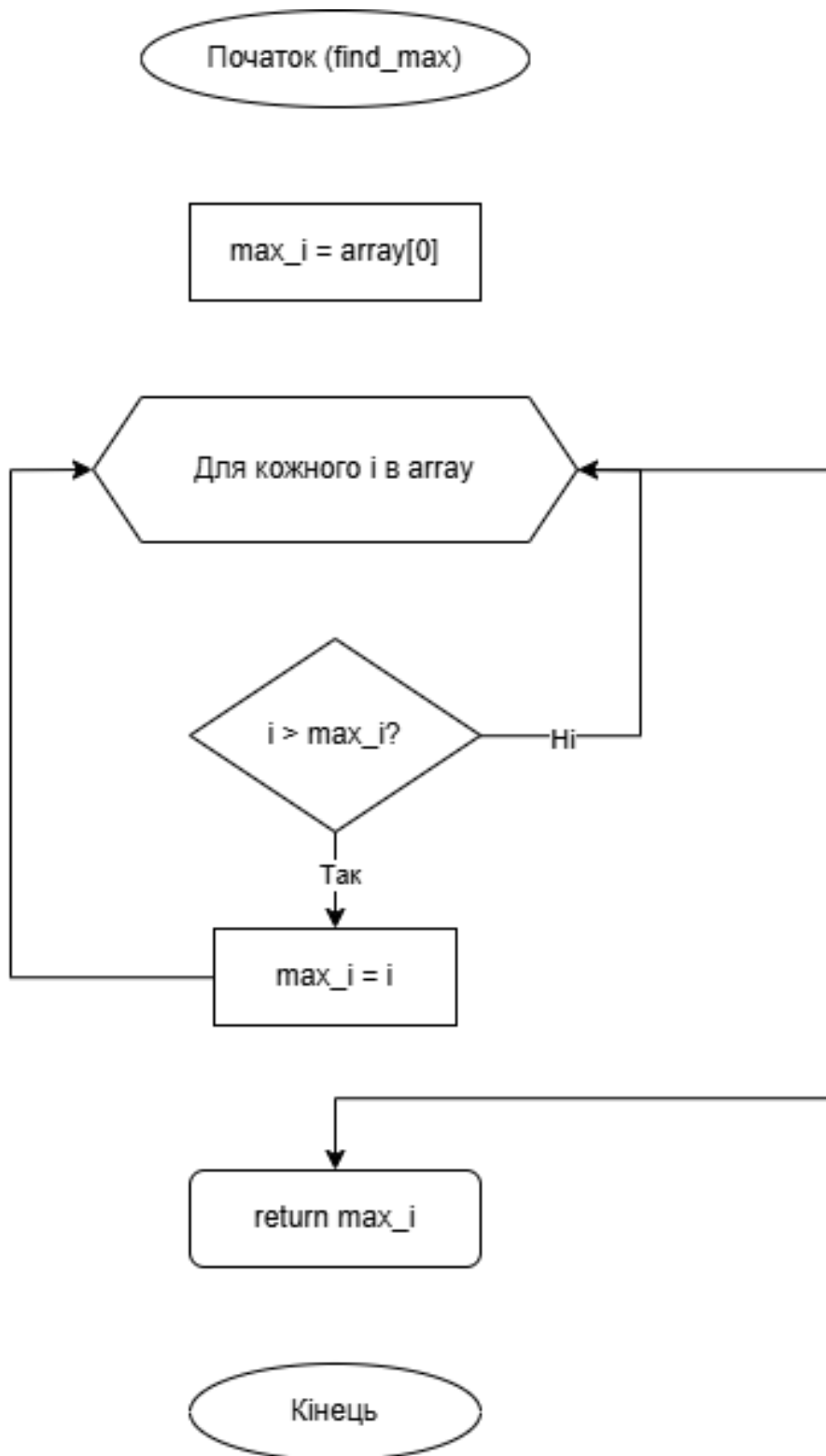


Рисунок 6 - Блок-схема знаходження максимального до завдання 2

Висновок:

В ході виконання лабораторної роботи #9 було опановано методи взаємодії з одновимірними масивами та запропоновані цікаві рішення заданих задач.