Міністерство освіти і науки України

Національний університет «Львівська політехніка»

Кафедра систем штучного інтелекту

**Звіт**

про виконання

**Лабораторних та практичних робіт № 4**

***з дисципліни:*** «Мови та парадигми програмування»

***з розділу***: «Прості структури даних. Одновимірні масиви. Двовимірні масиви. Алгоритми обробки»

***Виконала:***

студент групи ШІ-11

Дах Тарас Романович

## **Тема роботи:**

Прості структури даних. Одновимірні масиви. Двовимірні масиви. Алгоритми обробки

## **Мета роботи:**

## Ознайомитися з структурами даних та поглибити знання у працюванні з масивами. Практично застосувати знання, пишучи коди до практичних та лабораторних роботи.

## **Теоретичні відомості:**

### Теоретичні відомості з переліком важливих тем та індивідуальних план опрацювання їх:

#### Тема №1: Одновимірні масиви

* Джерела Інформації
  + [Урок №77. Масиви](https://acode.com.ua/urok-77-masyvy/)
  + [Урок №78. Фіксовані масиви](https://acode.com.ua/urok-78-fiksovani-masyvy/)
  + [Урок №79. Масиви і цикли](https://acode.com.ua/urok-79-masyvy-i-tsykly/)
  + [C++ Arrays](https://www.programiz.com/cpp-programming/arrays)
  + <https://www.youtube.com/watch?v=ULdbOaMBPYc&list=PLiPRE8VmJzOpn6PzYf0higmCEyGzo2A5g&index=41>
  + Матеріал з ВНС №4
* Що опрацьовано:
  + одновимірні масиви та їхнє застосування
* Статус: Ознайомлений
* Початок опрацювання теми: 28.11.2023
* Звершення опрацювання теми: 06.12.2023

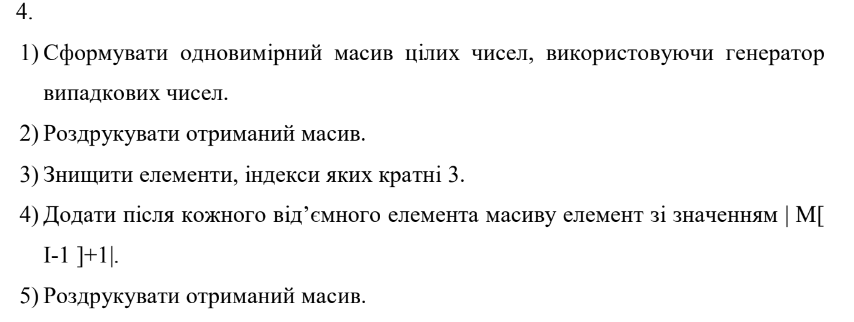
#### Тема №2: Двовимірні масиви

* Джерела Інформації
  + [Урок №81. Двовимірні масиви](https://acode.com.ua/urok-81-bagatovymirni-masyvy/)
  + [Урок №90. Динамічні масиви](https://acode.com.ua/urok-90-dynamichni-masyvy/)
  + [Multidimensional Arrays](https://www.programiz.com/cpp-programming/multidimensional-arrays)
  + [C++ Function and Array](https://www.programiz.com/cpp-programming/passing-arrays-function)
  + <https://www.youtube.com/watch?v=V2g3B9Zbh4Q&list=PLiPRE8VmJzOpn6PzYf0higmCEyGzo2A5g&index=42>
  + <https://www.youtube.com/watch?v=1DtZCv7xfb8&list=PLiPRE8VmJzOpn6PzYf0higmCEyGzo2A5g&index=43>
  + Матеріал у ВНС №5
* Що опрацьовано:
  + багатовимірні масиви та їхнє застосування
* Статус: Ознайомлений
* Початок опрацювання теми: 28.11.2023
* Звершення опрацювання теми: 06.12.2023

## **Виконання роботи:**

### 1. Опрацювання завдання та вимог до програм та середовища:

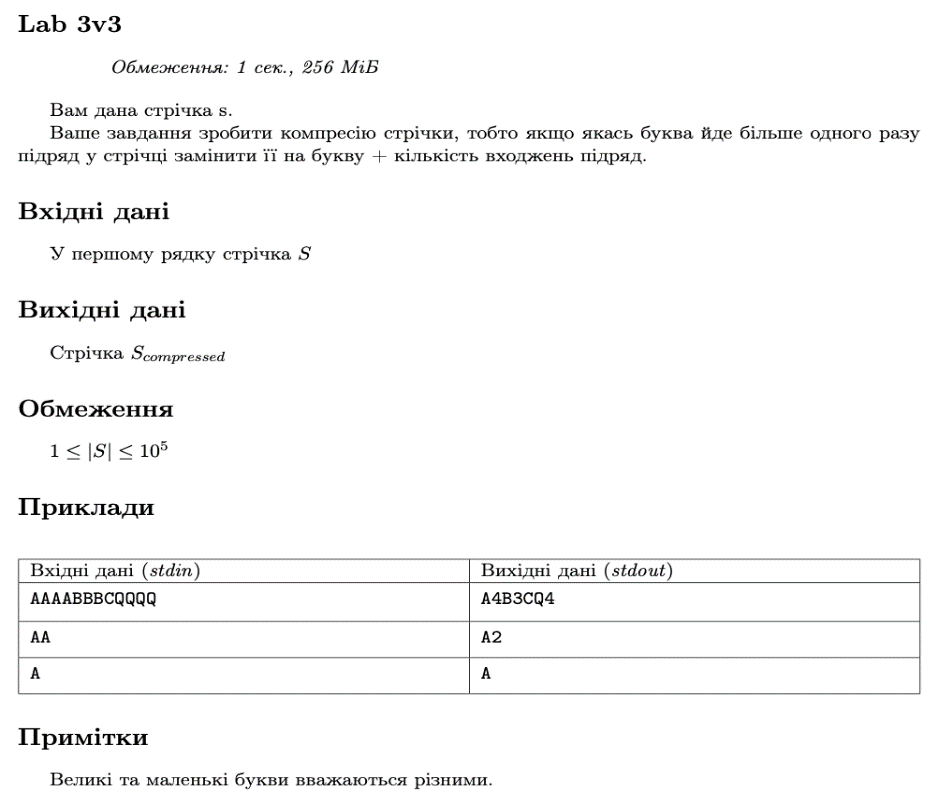
#### Завдання №1: VNS Lab 4



#### Завдання №2: VNS Lab 5

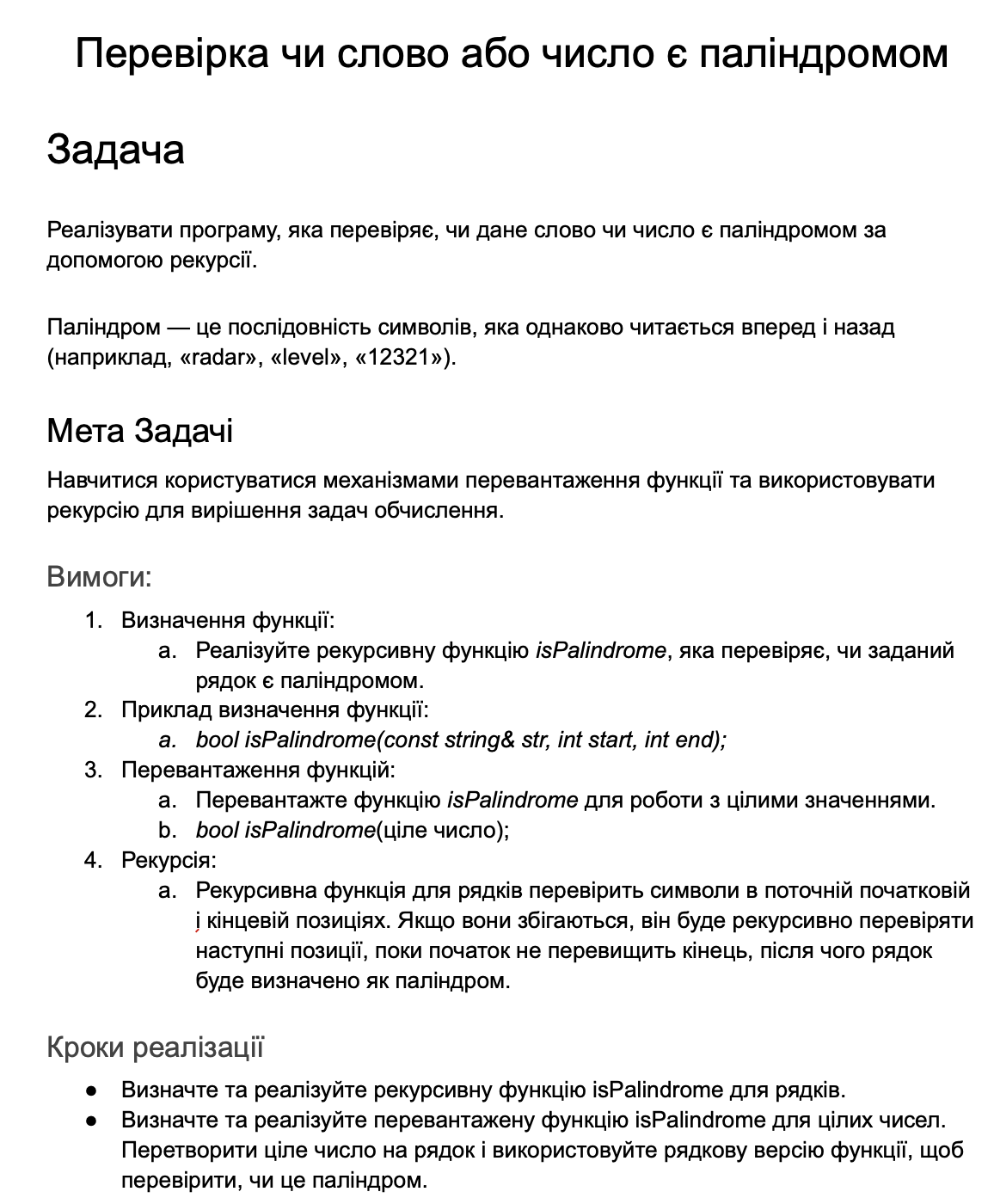
#### 

#### Завдання №3: Algotester Lab 2

Завдання №4: Algotester Lab 3

*Деталі до завдання №4*

#### Завдання №5: Class Practice

* Деталі завдання та важливі деталі для врахування в імплементації програми: 

*Деталі до завдання №5*

#### Завдання №6: Self Practice

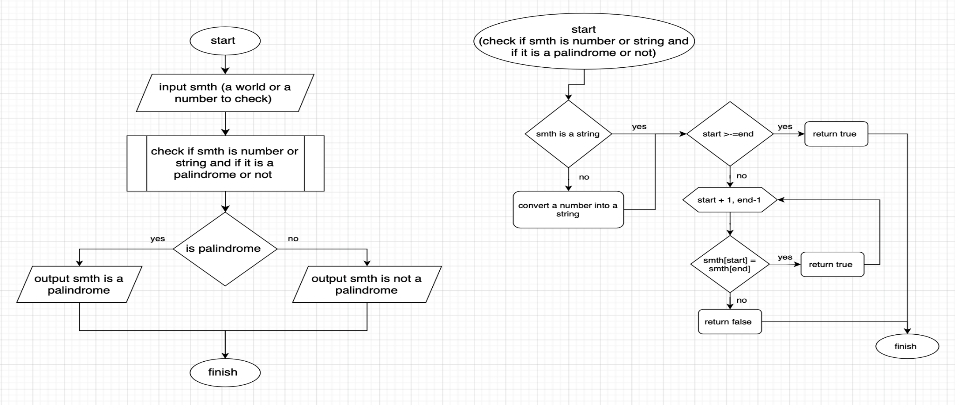
* Деталі завдання:

Створити одновимірний масив та функцію, яка знаходить значення суми елементів цього масиву

*Деталі до завдання №6*

### 2. Дизайн виконання завдань:

#### Програма № 5

* Блок-схема:

*Блок-схема до програми №5*

### 3. Код програм з посиланням на зовнішні ресурси:

#### Завдання №1: VNS Lab 4

#include <iostream>

#include <vector>

#include <random>

// Функція для створення одновимірного масиву цілих чисел за допомогою генератора випадкових чисел

std::vector<int> generateArray(int size) {

    std::vector<int> arr;

    std::random\_device rd;

    std::mt19937 gen(rd());

    std::uniform\_int\_distribution<> dis(-50, 49);

    for (int i = 0; i < size; ++i) {

        arr.push\_back(dis(gen)); // Генеруємо випадкові числа в діапазоні [-50, 49]

    }

    return arr;

}

// Функція для друку масиву

void printArray(const std::vector<int>& arr) {

    std::cout << "Масив:";

    for (int num : arr) {

        std::cout << " " << num;

    }

    std::cout << std::endl;

}

void removeElements(std::vector<int>& arr) {

    std::vector<int> newArr;

    for (int i = 0; i < arr.size(); ++i) {

        if ((i + 1) % 3 != 0) { // Видаляємо елементи, індекси яких кратні 3

            newArr.push\_back(arr[i]);

        }

    }

    arr = newArr;

}

// Функція для додавання після кожного від’ємного елемента масиву елементу зі значенням | M[i-1] + 1 |

void addAfterNegative(std::vector<int>& arr) {

    std::vector<int> newArr;

    for (int num : arr) {

        newArr.push\_back(num);

        if (num < 0) {

            newArr.push\_back(abs(newArr.back() + 1));

        }

    }

    arr = newArr;

}

int main() {

    // Генеруємо масив

    int arraySize = 10; // Вкажіть бажаний розмір масиву

    std::vector<int> myArray = generateArray(arraySize);

    // Виводимо початковий масив

    printArray(myArray);

    // Знищуємо елементи, індекси яких кратні 3

    removeElements(myArray);

    // Виводимо масив після знищення елементів

    printArray(myArray);

    // Додаємо після кожного від’ємного елемента масиву елемент зі значенням | M[i-1] + 1 |

    addAfterNegative(myArray);

    // Виводимо кінцевий масив

    printArray(myArray);

    return 0;

}

*Код до завдання №1*

#### Завдання №2: VNS Lab 5

#include <iostream>

#include <vector>

#include <cmath>

// Функція для обчислення скалярного добутку векторів

double dotProduct(const std::vector<double>& vec1, const std::vector<double>& vec2) {

    double result = 0.0;

    for (size\_t i = 0; i < vec1.size(); ++i) {

        result += vec1[i] \* vec2[i];

    }

    return result;

}

// Функція для перевірки ортонормованості матриці

bool isOrthonormalMatrix(const std::vector<std::vector<double>>& matrix) {

    size\_t rows = matrix.size();

    size\_t cols = matrix[0].size();

    // Перевірка розмірності та відмінності векторів

    if (rows != cols) {

        std::cerr << "Матриця не є квадратною. Вихід з функції." << std::endl;

        return false;

    }

    // Перевірка скалярних добутків

    for (size\_t i = 0; i < rows; ++i) {

        for (size\_t j = i + 1; j < rows; ++j) {

            if (dotProduct(matrix[i], matrix[j]) != 0.0) {

                std::cerr << "Скалярний добуток рядків " << i + 1 << " і " << j + 1 << " не дорівнює 0." << std::endl;

                return false;

            }

        }

    }

    // Перевірка скалярних добутків рядка на себе

    for (size\_t i = 0; i < rows; ++i) {

        if (std::abs(dotProduct(matrix[i], matrix[i]) - 1.0) > 1e-10) {

            std::cerr << "Скалярний добуток рядка " << i + 1 << " на себе не дорівнює 1." << std::endl;

            return false;

        }

    }

    // Якщо всі перевірки пройдені, матриця є ортонормованою

    return true;

}

int main() {

    // Введення розмірності матриці

    std::cout << "Введіть розмірність квадратної матриці: ";

    size\_t size;

    std::cin >> size;

    // Введення елементів матриці

    std::vector<std::vector<double>> inputMatrix(size, std::vector<double>(size, 0.0));

    std::cout << "Введіть елементи матриці:" << std::endl;

    for (size\_t i = 0; i < size; ++i) {

        for (size\_t j = 0; j < size; ++j) {

            std::cout << "Елемент [" << i + 1 << "][" << j + 1 << "]: ";

            std::cin >> inputMatrix[i][j];

        }

    }

    // Перевірка ортонормованості матриці

    if (isOrthonormalMatrix(inputMatrix)) {

        std::cout << "Матриця є ортонормованою." << std::endl;

    } else {

        std::cout << "Матриця не є ортонормованою." << std::endl;

    }

    return 0;

}

*Код до завдання №2*

#### Завдання №3: Algotester Lab 2

#include <iostream>

#include <vector>

using namespace std;

int main() {

    // Зчитуємо вхідні дані

    int N;

    cin >> N;

    vector<int> r(N);

    for (int i = 0; i < N; ++i) {

        cin >> r[i];

    }

    int a, b, c;

   cin >> a >> b >> c;

    // Видаляємо числа a, b, c з масиву

    for (int i = 0; i < N; ++i) {

        if (r[i] == a || r[i] == b || r[i] == c) {

            r.erase(r.begin() + i);

            --N;

            --i; // Не забуваємо зменшити індекс, оскільки елемент видаляється

        }

    }

    // Створюємо новий масив сум сусідніх елементів

    vector<int> sums;

    for (int i = 0; i < N - 1; ++i) {

        sums.push\_back(r[i] + r[i + 1]);

    }

    if (N > 1) {

        std::cout << N - 1 << std::endl;

        for (int sum : sums) {

            std::cout << sum << std::endl;

        }

    } else {

        std::cout << "0\n";

    }

    return 0;

}

*Код до завдання №3*

Завдання №4: Algotester Lab 3

#include <iostream>

#include <string>

std::string compressString(const std::string& s) {

    std::string compressed;

    int count = 1;

    for (size\_t i = 1; i <= s.size(); ++i) {

        if (i < s.size() && s[i] == s[i - 1]) {

            count++;

        } else {

            compressed += s[i - 1] + (count > 1 ? std::to\_string(count) : "");

            count = 1;

        }

    }

    return compressed;

}

int main() {

    std::string inputString;

    std::cin >> inputString;

    std::string compressedString = compressString(inputString);

    std::cout << compressedString << std::endl;

    return 0;

}

#### 

#### *Код до завдання №4*

#### Завдання №5: Class Practice

#include <iostream>

#include <string>

#include <algorithm>

// Рекурсивна функція для перевірки, чи рядок є паліндромом

bool isPalindrome(const std::string& str, int start, int end) {

    // Базовий випадок: якщо початок перевищив кінець, рядок є паліндромом

    if (start >= end) {

        return true;

    }

    // Рекурсивний крок: порівнюємо символи на початку та в кінці рядка

    if (str[start] != str[end]) {

        return false; // Якщо символи не співпадають, рядок не є паліндромом

    }

    // Рекурсивно перевіряємо наступні позиції

    return isPalindrome(str, start + 1, end - 1);

}

// Перевантажена функція для перевірки, чи ціле число є паліндромом

bool isPalindrome(int num) {

    // Перетворюємо ціле число на рядок

    std::string numStr = std::to\_string(num);

    // Викликаємо рядкову версію функції isPalindrome для перевірки паліндрому

    return isPalindrome(numStr, 0, numStr.length() - 1);

}

int main() {

    // Введення рядка для перевірки паліндрому

    std::cout << "Введіть рядок: ";

    std::string inputStr;

    std::getline(std::cin, inputStr);

    // Перевірка рядка

    if (isPalindrome(inputStr, 0, inputStr.length() - 1)) {

        std::cout << "\"" << inputStr << "\" є паліндромом." << std::endl;

    } else {

        std::cout << "\"" << inputStr << "\" не є паліндромом." << std::endl;

    }

    // Введення цілого числа для перевірки паліндрому

    std::cout << "Введіть ціле число: ";

    int inputNum;

    std::cin >> inputNum;

    // Перевірка цілого числа

    if (isPalindrome(inputNum)) {

        std::cout << inputNum << " є паліндромом." << std::endl;

    } else {

        std::cout << inputNum << " не є паліндромом." << std::endl;

    }

    return 0;

}

*Код до завдання №5*

#### Завдання №6: Self Practice

#include <iostream>

// Функція для обчислення суми елементів масиву

int calculateSum(const int\* arr, int size, int currentIndex) {

    if (currentIndex < size) {

        // Рекурсивно додаємо поточний елемент та переходимо до наступного

        return arr[currentIndex] + calculateSum(arr, size, currentIndex + 1);

    } else {

        return 0; // Базовий випадок: сума для порожнього масиву - 0

    }

}

int main() {

    int n;

    std::cout << "Enter the size of the array: ";

    std::cin >> n;

    int\* myArray = new int[n];

    std::cout << "Enter the elements of the array:" << std::endl;

    for (int i = 0; i < n; ++i) {

        std::cin >> myArray[i];

    }

    // Викликаємо функцію для обчислення суми елементів масиву

    int sum = calculateSum(myArray, n, 0);

    // Виводимо отриману суму

    std::cout << "Sum of array elements: " << sum << std::endl;

    // Звільнюємо виділену пам'ять

    delete[] myArray;

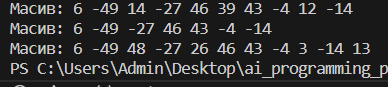
    return 0;

}

*Код до завдання №6*

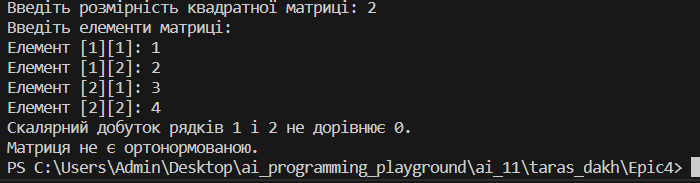
### 4. Результати виконання завдань, тестування та фактично затрачений час:

#### Завдання №1: VNS Lab 4



Час затрачений на виконання завдання: 1,5 год

#### Завдання №2: VNS Lab 5



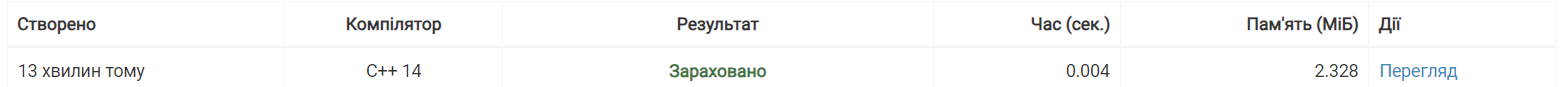
*Результати коду до завдання №2*

Час затрачений на виконання завдання: 1 год

#### Завдання №3: Algotester Lab 2

#### 

*Результати коду до завдання №3*



*Прийнято алготестером завдання*

Час затрачений на виконання завдання: 1 год

#### Завдання №4: Algotester Lab 3

#### 

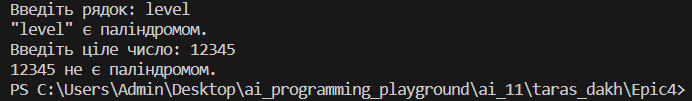
*Результати коду до завдання №4*

#### 

*Прийнято алготестером завдання*

Час затрачений на виконання завдання: 1.5 год

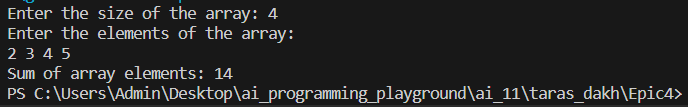
#### Завдання №5: Class Practice



*Результат коду до завдання №5*

Час затрачений на виконання завдання: 1 год

#### Завдання №6: Self Practice



*Результати коду до завдання №6*

Час затрачений на виконання завдання: 10 хв

## **Висновок:**

## На цій лабораторній роботі я дізнався про найпростіші структури даних, такі як одновимірні і двовимірні масиви, а також вектори. Крім цього я зрозумів їхній принцип і основні алгоритми щодо їх використання. Щоб закріпити вивчений матеріал нам було дано декілька програм з ВНС і Algotester.

ПР: