Міністерство освіти і науки України

Національний університет «Львівська політехніка»

Кафедра систем штучного інтелекту

Звіт

про виконання лабораторних та практичних робіт блоку № 4

На тему: «Одновимірні масиви. Двовимірні Масиви. Вказівники та Посилання. Динамічні масиви. Структури даних. Вкладені структури. Алгоритми обробки та робота з масивами та структурами.»

з дисципліни: «Основи програмування»

до:

ВНС Лабораторної Роботи №4

ВНС Лабораторної Роботи №5

Алготестер Лабораторної Роботи №2

Алготестер Лабораторної Роботи №3

Практичних Робіт до блоку №4

Виконала:

Студентка групи ШІ-13

Ходацька Аліна Віталіївна

Львів 2024

Тема роботи: Одновимірні масиви. Двовимірні Масиви. Вказівники та Посилання. Динамічні масиви. Структури даних. Вкладені структури. Алгоритми обробки та робота з масивами та структурами.

Мета роботи: Розібратися з різними видами масивів, також навчитися використовувати вказівники та зрозуміти що таке структури і які вони бувають.

Теоретичні відомості:

- масиви
- динамічні масиви
- структури
- вказівники

Джерела:

- C++ • Теорія • Урок 40 • Одновимірні масиви
- C++ • Теорія • Урок 41 • Багатовимірні масиви
- C++ • Теорія • Урок 42 • Масиви рядки
- C++ • Теорія • Урок 57 • Вказівники. Частина 1
- C++ Теорія • Урок 63 • struct

Виконання роботи

Завдання №1 VNS Lab 4 Task 1 Variant 12

1) Сформувати одновимірний масив цілих чисел, використовуючи генератор випадкових чисел.

2) Роздрукувати отриманий масив.

3) Поміняти місцями мінімальний і максимальний елементи масиву.

4) Знищити з масиву всі елементи, які перевищують його середнє значення більш, ніж на 10%.

5) Роздрукувати отриманий масив.

Запланований час: 30 хв

Витрачений час: 1 год

```cpp
#include <iostream>
#include <vector>
#include <cstdlib>
#include <ctime>

using namespace std;

int main() {
    // Крок 1: Створення масиву випадкових чисел
    srand(time(0));  // Ініціалізація генератора випадкових чисел
    int n = 10;  // Розмір масиву (можна змінити)
    vector<int> arr(n);

    for (int i = 0; i < n; i++) {
        arr[i] = rand() % 100;  // Заповнюємо масив випадковими числами від 0 до
99  }

    // Крок 2: Виведення початкового масиву
    cout << "Step 1: Initial array: ";
    for (int i = 0; i < n; i++) {
        cout << arr[i] << " ";
    }
    cout << endl;

    // Крок 3: Міняємо місцями мінімальний і максимальний елементи
    int minIndex = 0, maxIndex = 0;
    for (int i = 1; i < n; i++) {
        if (arr[i] < arr[minIndex]) {
            minIndex = i;
        }
        if (arr[i] > arr[maxIndex]) {
            maxIndex = i;
        }
    }
    // Міняємо місцями мінімальний і максимальний елементи
    swap(arr[minIndex], arr[maxIndex]);

    // Виводимо масив після зміни мінімального та максимального
    cout << "Step 2: After swapping min and max elements: ";
    for (int i = 0; i < n; i++) {
        cout << arr[i] << " ";
    }
    cout << endl;

    // Крок 4: Видаляємо елементи, що перевищують середнє значення на 10%
    int sum = 0;
    for (int i = 0; i < n; i++) {
        sum += arr[i];
    }
    double average = sum / (double)n;
    double threshold = average * 1.1;

    vector<int> filteredArr;
    for (int i = 0; i < n; i++) {
        if (arr[i] <= threshold) {
            filteredArr.push_back(arr[i]);
        }
    }

    // Виводимо масив після видалення елементів, що перевищують середнє на 10%
    cout << "Step 3: After removing elements greater than 10% above average: ";
    for (int i = 0; i < filteredArr.size(); i++) {
        cout << filteredArr[i] << " ";
    }
    cout << endl;

    return 0;
}
```

```
Microsoft Visual Studio Debu
Step 1: Initial array: 32 29 88 10 68 16 58 31 19 66
Step 2: After swapping min and max elements: 32 29 10 88 68 16 58 31 19 66
Step 3: After removing elements greater than 10% above average: 32 29 10 16 31 19
```

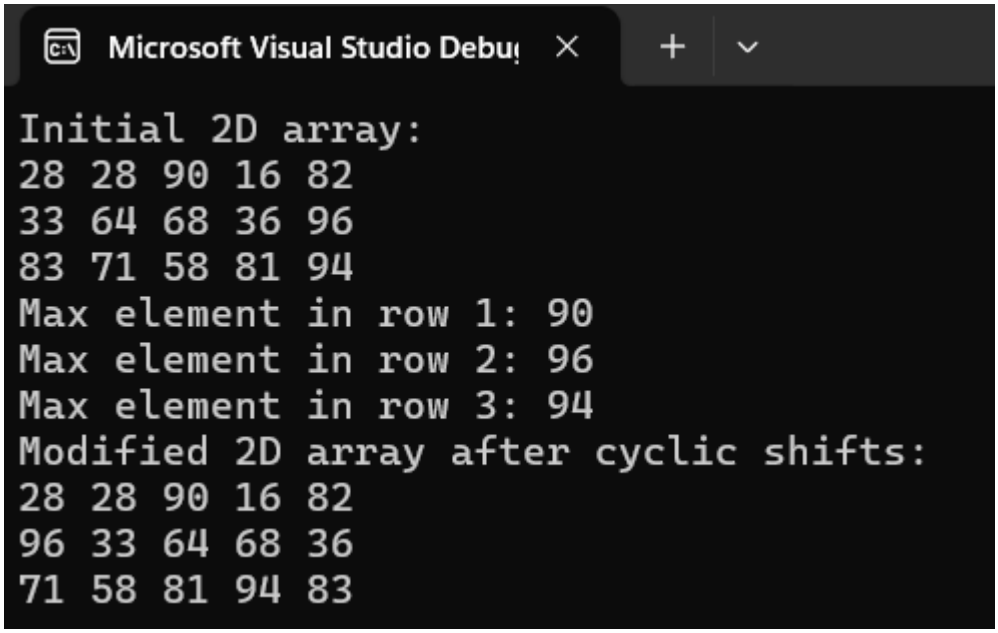
Завдання №2 VNS Lab 5 Task 1 Variant 12

Написати функцію, для пошуку максимального елемента в зазначеному рядку двовимірного масиву. Зсунути у двовимірному масиві всі рядки циклічно вправо на кількість елементів, яка дорівнює максимальному елементу в цьому рядку.

Запланований час: 30 хв

Витрачений час: 1,2 год

```cpp
#include <iostream>
#include <vector>

using namespace std;

// Функція для пошуку максимального елемента в рядку двовимірного масиву
int findMaxInRow(const vector<int>& row) {
    int maxElem = row[0];
    for (int i = 1; i < row.size(); i++) {
        if (row[i] > maxElem) {
            maxElem = row[i];
        }
    }
    return maxElem;
}

// Функція для циклічного зсуву елементів рядка вправо
void cyclicShiftRight(vector<int>& row, int shift) {
    int n = row.size();
    // Якщо зсув більший за кількість елементів, зменшуємо його
    shift = shift % n;
    vector<int> temp(n);

    // Копіюємо елементи в новий масив з циклічним зсувом
    for (int i = 0; i < n; i++) {
        temp[(i + shift) % n] = row[i];
    }

    // Переносимо елементи з тимчасового масиву назад в оригінальний рядок
    for (int i = 0; i < n; i++) {
        row[i] = temp[i];
    }
}

// Основна функція
int main() {
    // Створюємо двовимірний масив
    int rows = 3, cols = 5;  // кількість рядків і стовпців (можна змінити)
    vector<vector<int>> arr(rows, vector<int>(cols));

    // Ініціалізація масиву випадковими числами
    srand(time(0));
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            arr[i][j] = rand() % 100;  // Заповнюємо елементи випадковими числами від 0 до 99
        }
    }

    // Виведення початкового масиву
    cout << "Initial 2D array: " << endl;
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            cout << arr[i][j] << " ";
        }
        cout << endl;
    }

    // Для кожного рядка масиву знаходимо максимальний елемент і робимо циклічний зсув
    for (int i = 0; i < rows; i++) {
        int maxElem = findMaxInRow(arr[i]);  // Знайти максимальний елемент в рядку
        cout << "Max element in row " << i + 1 << ": " << maxElem << endl;

        cyclicShiftRight(arr[i], maxElem);  // Зсунути рядок вправо на maxElem
    }

    // Виведення зміненого масиву
    cout << "Modified 2D array after cyclic shifts: " << endl;
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            cout << arr[i][j] << " ";
        }
        cout << endl;
    }

    return 0;
}
```


Microsoft Visual Studio Debu
Initial 2D array:
28 28 90 16 82
33 64 68 36 96
83 71 58 81 94
Max element in row 1: 90
Max element in row 2: 96
Max element in row 3: 94
Modified 2D array after cyclic shifts:
28 28 90 16 82
96 33 64 68 36
71 58 81 94 83

Завдання №3 Algotester Lab 2 Variant 2

У вас є масив r розміром N. Також вам дано 3 цілих числа.

Спочатку ви маєте видалити з масиву ці 3 числа, які вам дані. Після цього перетворити цей масив у масив сум, розміром N_new − 1 (розмір нового масиву після видалення елементів), який буде відображати суми сусідніх елементів нового масиву.

Далі необхідно вивести масив сум на екран.

Input

У першому рядку ціле число N - кількість чисел

У другому рядку масив r, який складається з N цілих чисел

У третьому рядку 3 цілих числа, a, b, c, які треба видалити з масиву

Output

У першому рядку ціле число M - кількість чисел у масиві, який буде виведено

У наступному рядку M чисел - новий масив

Constraints

$1 \le N \le 10^5$

$0 \le r_i \le 10^5$

$0 \le a,b,c \le 10^5$

Запланований час: 30 хв

Витрачений час: 30 хв

```cpp
#include <iostream>
#include <vector>

using namespace std;

int main() {
    // Читання входу
    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;

    // Крок 1: Видалення елементів a, b, c
    vector<int> newArray;

    // Додаємо всі елементи, які не рівні a, b чи c
    for (int i = 0; i < N; i++) {
        if (r[i] != a && r[i] != b && r[i] != c) {
            newArray.push_back(r[i]);
        }
    }

    // Перевірка на кількість елементів після видалення
    int M = newArray.size();

    if (M < 2) {
        // Якщо елементів після видалення менше ніж 2, то неможливо створити масив сум
        cout << 0 << endl;
        return 0;
    }

    // Крок 2: Створення нового масиву сум
    vector<int> sumArray;
    for (int i = 0; i < M - 1; i++) {
        sumArray.push_back(newArray[i] + newArray[i + 1]);
    }

    // Крок 3: Виведення результату
    cout << sumArray.size() << endl;
    for (int i = 0; i < sumArray.size(); i++) {
        cout << sumArray[i] << " ";
    }
    cout << endl;

    return 0;
}
```

Завдання №4 Algotester Lab 3 Variant 3

Вам дана стрічка s.

Ваше завдання зробити компресію стрічки, тобто якщо якась буква йде більше одного разу підряд у стрічці замінити її на букву + кількість входжень підряд.

Input

У першому рядку стрічка S.

Output

Стрічка S_compressed.

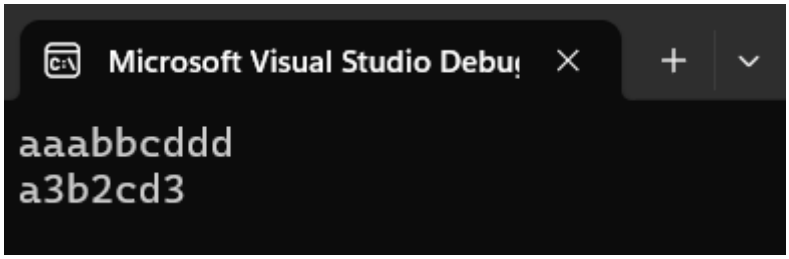
Constraints

1≤|S|≤10^5

Запланований час: 30 хв

Витрачений час: 25 хв

```cpp
#include <iostream>
#include <string>

using namespace std;

int main() {
    // Читання вхідної стрічки
    string s;
    cin >> s;

    // Якщо стрічка порожня (на всяк випадок, хоча обмеження гарантують, що довжина буде ? 1)
    if (s.empty()) {
        cout << 0 << endl;
        return 0;
    }

    // Змінна для зберігання результату компресії
    string compressed = "";

    int n = s.size();
    int i = 0;

    while (i < n) {
        char currentChar = s[i];
        int count = 1;

        // Підрахунок кількості однакових символів підряд
        while (i + 1 < n && s[i + 1] == currentChar) {
            count++;
            i++;
        }

        // Додавання символу та кількості повторів (якщо більше одного)
        compressed += currentChar;
        if (count > 1) {
            compressed += to_string(count);
        }

        // Переходимо до наступного символу
        i++;
    }

    // Виведення результату
    cout << compressed << endl;

    return 0;
}
```



Завдання №5 Practice work

Задача

Реалізувати програму, яка перевіряє, чи дане слово чи число є паліндромом за допомогою рекурсії.

Вимоги:

1. Визначення функції:

   a. Реалізуйте рекурсивну функцію *isPalindrome*, яка перевіряє, чи заданий рядок є паліндромом.

2. Приклад визначення функції:

   a. *bool isPalindrome (const string& str, int start, int end);*

3. Перевантаження функцій:

   a. Перевантажте функцію *isPalindrome* для роботи з цілими значеннями.

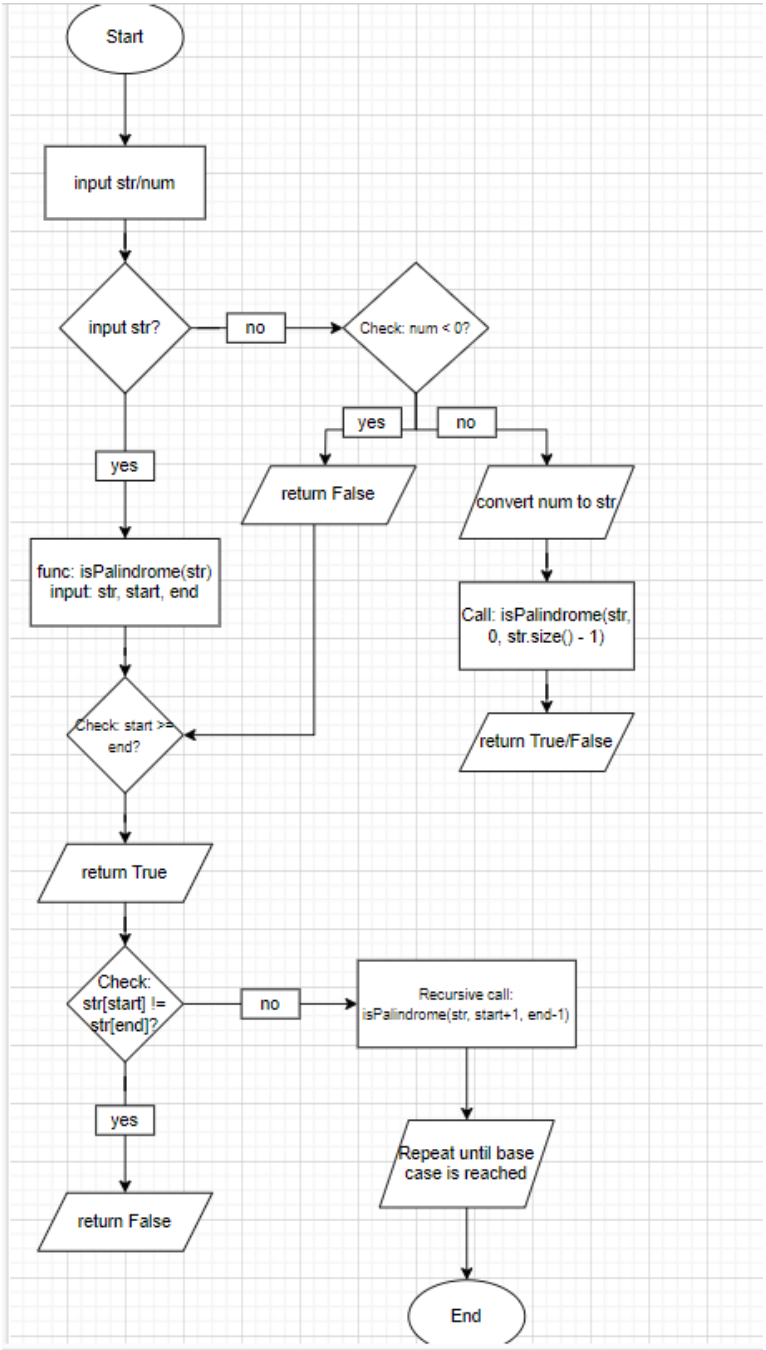
   b. *bool isPalindrome* (ціле число);

4. Рекурсія:

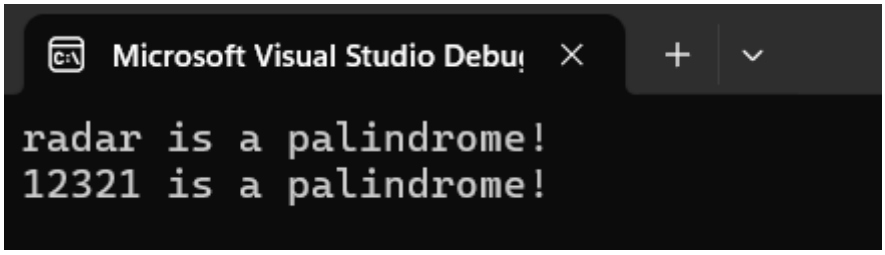
   a. Рекурсивна функція для рядків перевірить символи в поточній початковій і кінцевій позиціях. Якщо вони збігаються, буде рекурсивно перевіряти наступні позиції, поки початок не перевищить кінець, після чого рядок буде визначено як паліндром.

Запланований час: 50 хв

Витрачений час: 2 год


Start
input str/num
input str?
no
Check: num < 0?
yes
no
yes
return False
convert num to str
func: isPalindrome(str)
input: str, start, end
Call: isPalindrome(str,
0, str.size() - 1)
Check: start >=
end?
return True/False
return True
Check:
str[start] !=
str[end]?
no
Recursive call:
isPalindrome(str, start+1, end-1)
yes
Repeat until base
case is reached
return False
End

```cpp
#include <iostream>
#include <string>
using namespace std;

//Рекурсивна функція для перевірки, чи є рядок паліндромом
bool isPalindrome(const string& str, int start, int end) {
    // Базовий випадок: коли індекс start більше або дорівнює end
    if (start >= end) {
        return true;
    }

    // Якщо символи на початку і в кінці не співпадають
    if (str[start] != str[end]) {
        return false;
    }

    // Рекурсивно перевіряємо наступні символи
    return isPalindrome(str, start + 1, end - 1);
}

// Функція для перевірки, чи є число паліндромом
bool isPalindrome(int num) {
    // Якщо число негативне, воно не може бути паліндромом
    if (num < 0) return false;

    // Перетворюємо число в рядок і викликаємо функцію для рядка
    string str = to_string(num);
    return isPalindrome(str, 0, str.size() - 1);
}

int main() {
    // Приклад перевірки для рядка
    string word = "radar";
    if (isPalindrome(word, 0, word.size() - 1)) {
        cout << word << " is a palindrome!" << endl;
    }
    else {
        cout << word << " is not a palindrome!" << endl;
    }

    // Приклад перевірки для числа
    int number = 12321;
    if (isPalindrome(number)) {
        cout << number << " is a palindrome!" << endl;
    }
    else {
        cout << number << " is not a palindrome!" << endl;
    }

    return 0;
}
```



Завдання №6 Self Practice work Algotester Lab 3 Variant 1

Ви з’явились у світі під назвою Атод посеред Пустелі Безправ’я. Так сталось, що Ви попали саме в той час і місце, де ведеться битва між чаклункою Ліною і темними силами, які хочуть знищити цей світ. На жаль, трапилась халепа, бо деякі слова із книги чар були пошкоджені під час битви. Одне таке слово можна відновити виконавши ритуал зцілення над пошкодженими буквами. Ритуал зцілення можна виконати на всіх підряд розташованих пошкоджених буквах. Вам не залишається нічого іншого як допомогти Ліні відновити ці слова і сказати скільки мінімально треба провести таких ритуалів, щоб прочитати одне з наймогутніших у цьому світі заклять - Поневолення Дракона!

визначити мінімальну кількість ритуалів зцілення, необхідних для відновлення всіх пошкоджених слів у заклятті

Input

У першому рядку N - кількість рядків у заклятті.

В наступних N рядках - набір слів w_1,…, w_M, розділених пробілами, де кожне слово може містити малі латинські літери та символ #, який позначає пошкоджену букву.

Output

Єдине ціле число - мінімальна кількість ритуалів, які потрібно провести, щоб відновити закляття.

Constraints

1<=N<=10^3

1<=M<=42

1<=||w_i||<=42

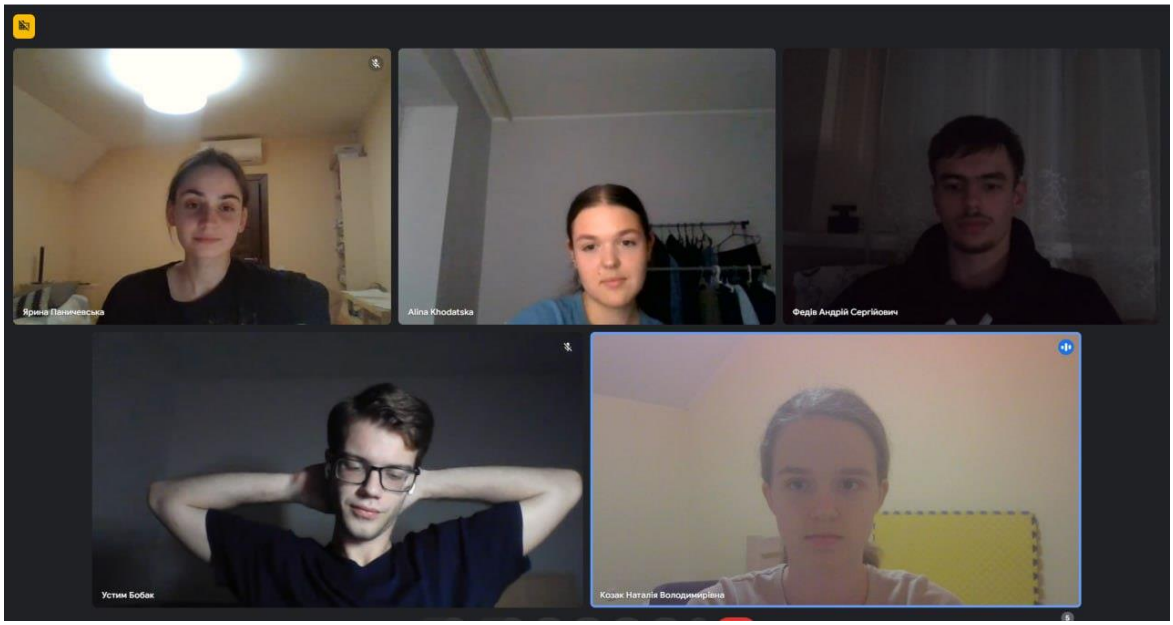
Запланований час: 30 хв

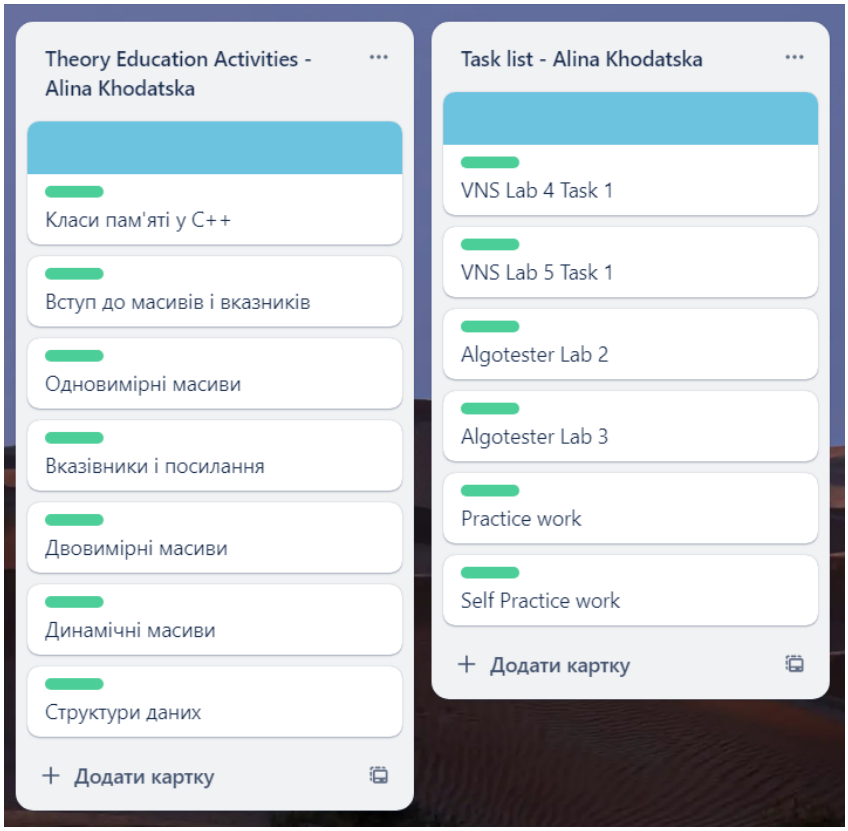
Витрачений час: 40 хв

```cpp
#include <iostream>
#include <vector>
#include <sstream>

using namespace std;

int main() {
    // Зчитуємо кількість рядків
    int N;
    cin >> N;
    cin.ignore();  // ігноруємо переведення на новий рядок після N

    int totalRituals = 0;

    // Обробляємо кожен рядок
    for (int i = 0; i < N; ++i) {
        string line;
        getline(cin, line);

        stringstream ss(line);
        string word;

        // Обробляємо кожне слово в рядку
        while (ss >> word) {
            int rituals = 0;
            bool inRitual = false;

            // Перевіряємо кожен символ в слові
            for (char c : word) {
                if (c == '#') {
                    if (!inRitual) {
                        // Початок нової послідовності пошкоджених символів
                        rituals++;
                        inRitual = true;
                    }
                }
                else {
                    // Завершення послідовності пошкоджених символів
                    inRitual = false;
                }
            }

            totalRituals += rituals;
        }
    }

    // Виводимо загальну кількість ритуалів
    cout << totalRituals << endl;

    return 0;
}
```

```
Microsoft Visual Studio Debu
3
hello #world
this ##is#a#test
###magic##spell###
7
```

Зустріч з командою та дошка в Trello





Висновок: У ході виконання роботи було досліджено та вивчено основи роботи з одновимірними та двовимірними масивами, динамічними масивами, вказівниками, посиланнями, а також структурами даних, включаючи вкладені структури. Було розглянуто алгоритми обробки масивів та методи їх використання для вирішення практичних завдань.

Посилання на pull request: https: //github.com/artificial-intelligence-department/ai_programming_playground_2024/pull/404