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-12
Лазаревич Юлія Дмитрівна

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:

Ознайомитись з одновимірними, двовимірними та динамічними масивами, вказівниками та посиланнями, структурами даних, вкладеними структурами та алгоритмами обробки.

Теоретичні відомості:

Тема №1: Класи пам'яті у C++.

- Джерела Інформації:
  - Класи, Об’єкти і Методи в C++
- Що опрацьовано:
  - Статична пам’ять – область пам'яті, де зберігаються глобальні та статичні змінні, які ініціалізуються на етапі компіляції та які існують протягом усього виконання програми. (static int counter = 0;)
  - Динамічна пам’ять виділяється під час виконання програми за допомогою new і звільняється з використанням delete, що дозволяє створювати змінні або масиви з довжиною, яка визначається під час виконання програми.(int* arr = new int[5]; delete[] arr;)
  - Стек використовується для зберігання локальних змінних і викликів функцій. Він має обмежену пам’ять і працює за принципом "останнім увійшов — першим вийшов" (LIFO).
  - Виділення пам’яті здійснюється через new або malloc, а вивільнення — через delete або free. Неправильне управління може призвести до витоків пам'яті.
- Статус: ознайомлена.
- Початок опрацювання теми: 20.11.24
- Завершення опрацювання теми 25.11.24

Тема №2: Одновимірні та двовимірні масиви.

- Джерела Інформації:
  - Що таке масив у програмуванні: основи та застосування
- Що опрацьовано:
  - Одновимірні:

- Створення та ініціалізація: int arr[5] = {1, 2, 3, 4, 5};
- Основні операції: Індексація (arr[2]), присвоєння (arr[0] = 10), читання.
- Цикли для обходу: for (int i = 0; i < 5; i++) {std::cout << arr[i];}
- Функції: Масиви передаються як вказівники.
- Алгоритми сортування: Наприклад, сортування вставками.

▪ Двовимірні(представляють таблицю елементів (матрицю)):

- Створення та ініціалізація: int matrix[3][4] = { {1, 2, 3, 4}, {5, 6, 7, 8}, {9, 10, 11, 12} };
- Вкладені цикли для обходу:

```
for (int i = 0; i < 3; i++) {
    for (int j = 0; j < 4; j++) {
        std::cout << matrix[i][j] << " ";
    }
    std::cout << std::endl;
}
```

- На практиці двовимірні масиви корисні для задач обробки матриць, наприклад, знаходження суми елементів рядка чи стовпця.
- Масиви передаються через вказівники:

```
void printMatrix(int matrix[][4], int rows) {
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < 4; j++) {
            std::cout << matrix[i][j] << " ";
        }
        std::cout << std::endl;
    }
}
```

- Статус: ознайомлена.
- Початок опрацювання теми: 20.11.24
- Завершення опрацювання теми 25.11.24

Тема №3: Вказівники та Посилання.

- Джерела Інформації:
  - Урок №92. Посилання
  - Урок №84. Вказівники
- Що опрацьовано:
  - Вказівники для доступу до масивів:

```
int arr[] = {1, 2, 3};
int* p = arr;
```

  - Арифметика вказівників: Наприклад, p + 1 вказує на наступний елемент.
  - Різниця між вказівниками та посиланнями: Посилання є статичними, а вказівники можна змінювати.
  - Динамічне виділення пам’яті:

```
int* arr = new int[10];
delete[] arr;
```

- Статус: ознайомлена.
- Початок опрацювання теми: 27.11.24
- Завершення опрацювання теми 28.11.24

Тема №4: Динамічні Масиви.

- Джерела Інформації:
  - Урок №90. Динамічні масиви
- Що опрацьовано:
  - Масив створюється під час виконання за допомогою оператора new:
    ```
    int* arr = new int[10];
    ```
  - Завжди потрібно вивільняти пам’ять:
    ```
    delete[] arr;
    ```
  - Для зміни розміру можна створити новий масив, копіювати вміст та видалити старий:
    ```
    int* newArr = new int[newSize];
    std::copy(oldArr, oldArr + oldSize, newArr);
    delete[] oldArr;
    ```
  - Передаються у функції як вказівники разом із розміром.
- Статус: ознайомлена.
- Початок опрацювання теми: 28.11.24
- Завершення опрацювання теми 29.11.24

Тема №5: Структури Даних

- Джерела Інформації:
  - Урок №64. Структури
- Що опрацьовано:
  - Оголошення та використання:
    ```
    struct Point {
       int x, y;
    };
    Point p = {10, 20};
    ```
  - Масиви у структурах:
    ```
    struct Student {
       std::string name;
       int grades[5];
    };
    ```
  - Структури передаються у функції за значенням або посиланням:
    ```
    void printPoint(const Point& p) {
       std::cout << p.x << ", " << p.y;
    }
    ```

- Union – тип даних, де всі поля ділять одну область пам'яті:

```
union Data {
   int i;
   float f;
};
```

- Переліки:

```
enum Color { RED, GREEN, BLUE };
```

- Статус: ознайомлена.
- Початок опрацювання теми: 28.11.24
- Завершення опрацювання теми 29.11.24

Тема №6: Вкладені структури та їх використання.

- Джерела Інформації:
  - Урок №64. Структури
- Що опрацьовано:
  - Оголошення:

```
struct Address {
   std::string city;
   std::string street;
};

struct Person {
   std::string name;
   Address address;
};
```

  - Передача у функції:

```
void printPerson(const Person& p) {
   std::cout << p.name << ", " << p.address.city;
}
```

  - Застосування: Використовуються для моделювання складних об’єктів, таких як бази даних.
- Статус: ознайомлена.
- Початок опрацювання теми: 12.10.24
- Завершення опрацювання теми 22.10.24

Тема №7: Алгоритми обробки та робота з масивами та структурами.

- Джерела Інформації:
  - Урок №77. Масиви
- Що опрацьовано:
  - Сортування: Наприклад, сортування бульбашкою:

```
void bubbleSort(int arr[], int n) {
   for (int i = 0; i < n - 1; i++) {
      for (int j = 0; j < n - i - 1; j++) {
```

```
                if (arr[j] > arr[j + 1]) {
                    std::swap(arr[j], arr[j + 1]);
                }
            }
        }
    }
```

- Пошук: Лінійний або бінарний пошук у масиві.
- Маніпуляції зі структурами: Наприклад, обчислення середнього балу студента з використанням масиву оцінок

- Статус: ознайомлена.
- Початок опрацювання теми: 12.10.24
- Завершення опрацювання теми 22.10.24

Виконання роботи:

1. Опрацювання завдання та вимог до програм та середовища:

Завдання №1 – VNS Lab 4 – Task 1 – Variant 10.

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

2) Роздрукувати отриманий масив.

3) Знищити 5 перших елементів масиву.

4) Додати в кінець масиву 3 нових елементи.

5) Роздрукувати отриманий масив.

Методичні вказівки:

1) При виконанні роботи використовуються статичні масиви. Для організації статичних масивів із псевдозмінними межами необхідно оголосити масив досить великої довжини, наприклад, 100 елементів: int N=100; int a[N];

Потім користувач вводить реальну довжину масиву (не більше N) і працює з масивом тієї довжини, що він сам вказав. Інші елементи (хоча пам'ять під них і буде виділена) не розглядаються.

2) При зменшенні або збільшенні довжини масиву необхідно змінювати його реальну довжину.

Завдання №2 – VNS Lab 5 – Task 1 – Variant 10.

Використовуючи функції, розв’язати зазначене у варіанті завдання. Масив

повинен передаватися у функцію як параметр.

Написати функцію, що перевіряє чи є від'ємні елементи в зазначеному рядку двовимірного масиву. Знищити з масиву всі рядки з від'ємними елементами, знищений рядок заповнюється 0 і переноситься в кінець масиву.

Завдання №3 – Algotester Lab 2 – Variant 3.

Вам дано масив цілих чисел розміром N, на першій та останній клітинці розміщено по дрону. Вони одночасно взлітають.

На початку кожного ходу швидкість дрону стає рівною значенню клітинки, у якій він знаходиться. Тобто лівий дрон у першу секунду з клітинки з індексом 1 перелетить у клітинку з індексом a1, тобто його наступна позиція рахується як поточна позиція + число у поточній позиції (перегляньте пояснення для візуалізації) Правий робить аналогічно в протилежну сторону.

Вони це роблять до моменту, коли трапиться одна з зазначених подій:

Якшо 2 дрони опиняються в одній клітинці - ви виводите Collision.

Якщо лівий дрон опиниться справа від правого - це Miss

У випадку якщо вони зупиняться один навпроти одного, тобто у клітинках ai та ai+1 – виведіть «Stopped» Врахуйте, що перевіряти треба також до взльоту.

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.

Вхідні дані

У першому рядку стрічка S

Вихідні дані

Стрічка Scompressed

Обмеження

$1 \leq |S| \leq 105$

Завдання №5 – Class Practice Work – Перевірка чи слово або число є паліндромом.

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

Паліндром — це послідовність символів, яка однаково читається вперед і назад (наприклад, «radar», «level», «12321»).

Мета Задачі

Навчитися користуватися механізмами перевантаження функції та використовувати рекурсію для вирішення задач обчислення.

Вимоги:

1. Визначення функції:

1. Реалізуйте рекурсивну функцію *isPalindrome*, яка перевіряє, чи заданий рядок є паліндромом.
2. Приклад визначення функції:

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

3. Перевантаження функцій:

1. Перевантажте функцію *isPalindrome* для роботи з цілими значеннями.
2. *bool isPalindrome*(ціле число);

4. Рекурсія:

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

Кроки реалізації

- Визначте та реалізуйте рекурсивну функцію isPalindrome для рядків.
- Визначте та реалізуйте перевантажену функцію isPalindrome для цілих чисел. Використати математичний підхід щоб перевірити чи число є паліндромом.

Завдання №6 - Self Practice Work – Algotester Lab 3 – Variant 2.

Вам дано 2 масиви розміром N та M. Значення у цих масивах унікальні.

Ваше завдання вивести у першому рядку кількість елементів, які наявні в обох масивах одночасно, у другому кількість унікальних елементів в обох масивах разом.

Вхідні дані

У першому рядку ціле число NN

у другому рядку NN цілих чисел a1..ana1..an

У третьому рядку ціле число MM

у четвертому рядку MM цілих чисел b1..bnb1..bn

Вихідні дані

У першому рялку одне ціле число - кількість елементів, які наявні в обох масивах одночасно.

У другому рядку кількість унікальних елементів в обох масивах (тобто кількість унікальних елементів у масиві, який буде об'єднанням двох даних).

Додаткове завдання №7 – 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!

Вхідні дані

У першому рядку NN - кількість рядків у заклятті.

В наступних NN рядкях - набір слів w1,…,wMw1,…,wM, розділених пробілами, де кожне слово може містити малі латинські літери та символ ##, який позначає пошкоджену букву.

Вихідні дані

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

2. **Код програм з посиланням на зовнішні ресурси:**

Завдання №1 – VNS Lab 4 – Task 1 – Variant 10.

vns_lab_4_task_john_black.cpp > ...

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

using namespace std;

int main() {
    const int MAX_SIZE = 100;
    int array[MAX_SIZE];
    int realLength;

    cout << "Enter the size of the array (max " << MAX_SIZE << "): ";
    cin >> realLength;

    if (realLength > MAX_SIZE || realLength <= 0) {
        cout << "Invalid array size!" << endl;
        return 1;
    }

    srand(time(0));
    for (int i = 0; i < realLength; i++) {
        array[i] = rand() % 100000;
    }

    cout << "Initial array:" << endl;
    for (int i = 0; i < realLength; i++) {
        cout << array[i] << " ";
    }
    cout << endl;

    if (realLength > 5) {
        for (int i = 0; i < realLength - 5; i++) {
            array[i] = array[i + 5];
        }
        realLength -= 5;
    } else {
        realLength = 0;
    }

    int newElements[3];
    for (int i = 0; i < 3; i++) {
        newElements[i] = rand() % 100000;
        array[realLength + i] = newElements[i];
    }
    realLength += 3;

    cout << "Resulting array:" << endl;
    for (int i = 0; i < realLength; i++) {
        cout << array[i] << " ";
    }
    cout << endl;

    return 0;
}
```

```
Enter the size of the array (max 100): 10
Initial array:
14446 13376 30504 16260 22187 5688 1910 10040 9201 13945
Resulting array:
5688 1910 10040 9201 13945 20156 6289 15523
```

Завдання №2 – VNS Lab 5 – Task 1 – Variant 10.

vns_lab_5_task_10_yuliia_lazarevych.cpp > ...

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

//використовую функцію для перевірки наявності від'ємних елементів у рядку
//масив передається за посиланням як параметр, тому використовуємо & в ньому, що дозволить працювати з реальним масивом, а не з копією.
bool hasNegative(int (&arr)[5], int cols) {
    for (int i = 0; i < cols; i++) {
        if (arr[i] < 0) {
            return true;
        }
    }
    return false;
}

//використовую функцію для обробки масиву, яка замінює рядки з від'ємними елементами на нулі
//передаю масив за посиланням, щоб зміни в ньому були збережені, заміна рядка і його переміщення здійснюється в реальному масиві, а не в його копії.
void processArray(int (&arr)[5][5], int rows, int cols) {
    for (int i = 0; i < rows; i++) {
        if (hasNegative(arr[i], cols)) {
            for (int j = 0; j < cols; j++) {
                arr[i][j] = 0;
            }
            for (int k = i; k < rows - 1; k++) {
                for (int j = 0; j < cols; j++) {
                    arr[k][j] = arr[k + 1][j];
                }
            }
        }
    }
}

//використовую функцію для виведення масиву на екран
void printArray(int (&arr)[5][5], int rows, int cols) {
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            cout << arr[i][j] << " ";
        }
        cout << endl;
    }
}

int main() {
    // ініціалізую двовимірний масив
    int arr[5][5] = {
        {1, 2, 3, 4, -1},
        {6, 7, 8, 9, 10},
        {-1, -2, -3, -4, -5},
        {11, 12, 13, 14, 15},
        {16, 17, 18, 19, 20}
    };

    int rows = 5;
    int cols = 5;

    cout << "Масив до обробки:" << endl;
    printArray(arr, rows, cols);

    processArray(arr, rows, cols);

    cout << "Масив після обробки:" << endl;
    printArray(arr, rows, cols);

    return 0;
}
```

```
Масив до обробки:
1 2 3 4 -1
6 7 8 9 10
-1 -2 -3 -4 -5
11 12 13 14 15
16 17 18 19 20
Масив після обробки:
6 7 8 9 10
11 12 13 14 15
16 17 18 19 20
16 17 18 19 20
16 17 18 19 20
```

Завдання №3 – Algotester Lab 2 – Variant 3.

algotester_lab_2_task_yuliia_lazarevych.cpp > main()

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

//використовую структуру для зберігання інформації про дрон
struct Drone {
    int position;
};

int main() {
    int N;
    cin >> N;

    //використовую динамічний масив для зберігання швидкостей
    int* arr = new int[N + 1];
    for (int i = 1; i <= N; ++i) {
        cin >> arr[i];
    }

    //ініціалізую дрони
    Drone leftDrone = {1};
    Drone rightDrone = {N};

    while (true) {
        if (leftDrone.position == rightDrone.position) {
            cout << leftDrone.position << " " << rightDrone.position << endl;
            cout << "Collision" << endl;
            break;
        }
        if (leftDrone.position + 1 == rightDrone.position) {
            cout << leftDrone.position << " " << rightDrone.position << endl;
            cout << "Stopped" << endl;
            break;
        }
        if (leftDrone.position > rightDrone.position) {
            cout << leftDrone.position << " " << rightDrone.position << endl;
            cout << "Miss" << endl;
            break;
        }

        leftDrone.position += arr[leftDrone.position];
        rightDrone.position -= arr[rightDrone.position];
    }

    //звільненюю динамічну пам'ять
    delete[] arr;

    return 0;
}
```

Створено	Компілятор	Результат	Час (сек.)	Пам'ять (МіБ)	Дії
декілька секунд тому	C++ 23	Зараховано	0.003	1.219	Перегляд

Завдання №4 – Algotester Lab 3 – Variant 3.

algotester_lab_3_task_3_yuliia_lazarevych.cpp > ...

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

using namespace std;

//використовую структуру для зберігання символу та кількості його входжень
struct CharCount {
    char character;
    int count;
};

//використовую функцію для компресії стрічки
string compressString(const string& s) {
    vector<CharCount> compressed; //використовую динамічний масив для зберігання структури CharCount

    //ітерую через стрічку
    for (size_t i = 0; i < s.length(); ++i) {
        if (compressed.empty() || compressed.back().character != s[i]) {
            compressed.push_back({s[i], 1});
        } else {
            compressed.back().count++;
        }
    }

    string result;
    for (const auto& entry : compressed) {
        result += entry.character;
        if (entry.count > 1) {
            result += to_string(entry.count);
        }
    }

    return result;
}

int main() {
    string s;
    cin >> s;
    string compressed = compressString(s);
    cout << compressed << endl;
    return 0;
}
```

Створено	Компілятор	Результат	Час (сек.)	Пам'ять (МіБ)	Дії
декілька секунд тому	C++ 23	Зараховано	0.003	1.352	Перегляд

Завдання №5 – Class Practice Work – Перевірка чи слово або число є паліндромом.

practice_work_team_yuliia_lazarevych.cpp > ...

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

using namespace std;

//використовую структуру для зберігання інформації про перевірку
struct Data {
    string value;
    bool isPalindrome;
};

//використовую вкладену структуру для групування даних про рядки і числа
struct TestData {
    vector<Data> strings; //використовую масив даних для рядків
    vector<Data> numbers; //та масив даних для чисел
};

//використовую рекурсивну функцію для перевірки рядка на паліндромність (АЛГОРИТМ ОБРОБКИ)
bool isPalindrome(const string& str, int start, int end) {
    if (start >= end) {
        return true;
    }
    if (str[start] != str[end]) {
        return false;
    }
    return isPalindrome(str, start + 1, end - 1);
}

bool isPalindrome(int number) {
    string str = to_string(number);
    return isPalindrome(str, 0, str.size() - 1);
}

//використовую функцію для перевірки масивів даних на паліндромність
TestData checkPalindromes(const vector<string>& stringData, const vector<int>& numberData) {
    TestData results; // ВИКОРИСТАННЯ ВКЛАДЕНИХ СТРУКТУР

    //використовую обробку рядків (АЛГОРИТМ ОБРОБКИ МАСИВУ РЯДКІВ)
    for (const auto& str : stringData) {
        results.strings.push_back({str, isPalindrome(str, 0, str.size() - 1)});
    }

    //використовую обробку чисел (АЛГОРИТМ ОБРОБКИ МАСИВУ ЧИСЕЛ)
    for (int num : numberData) {
        results.numbers.push_back({to_string(num), isPalindrome(num)});
    }

    return results;
}

//використовую функцію для виведення результатів (ВИКОРИСТАННЯ СТРУКТУР ДЛЯ ВИВЕДЕННЯ)
void printResults(const TestData& data) {
    cout << "Результати перевірки рядків:" << endl;
    for (const auto& strData : data.strings) {
        cout << "Рядок \"" << strData.value << "\" є паліндромом? "
             << (strData.isPalindrome ? "Так" : "Ні") << endl;
    }

    cout << "\nРезультати перевірки чисел:" << endl;
    for (const auto& numData : data.numbers) {
        cout << "Число " << numData.value << " є паліндромом? "
             << (numData.isPalindrome ? "Так" : "Ні") << endl;
    }
}

int main() {
    //використовую масиви для введення текстових данних
    vector<string> stringData = {"radar", "hello", "level", "world"}; //масив рядків
    vector<int> numberData = {12321, 12345, 1112111, 98789};          //масив чисел

    //перевіряю паліндромів завдяки алгоритму обробки
    TestData results = checkPalindromes(stringData, numberData);

    printResults(results);

    return 0;
}
```

```
Результати перевірки рядків:
Рядок "radar" є паліндромом? Так
Рядок "hello" є паліндромом? Ні
Рядок "level" є паліндромом? Так
Рядок "world" є паліндромом? Ні

Результати перевірки чисел:
Число 12321 є паліндромом? Так
Число 12345 є паліндромом? Ні
Число 1112111 є паліндромом? Так
Число 98789 є паліндромом? Так
```

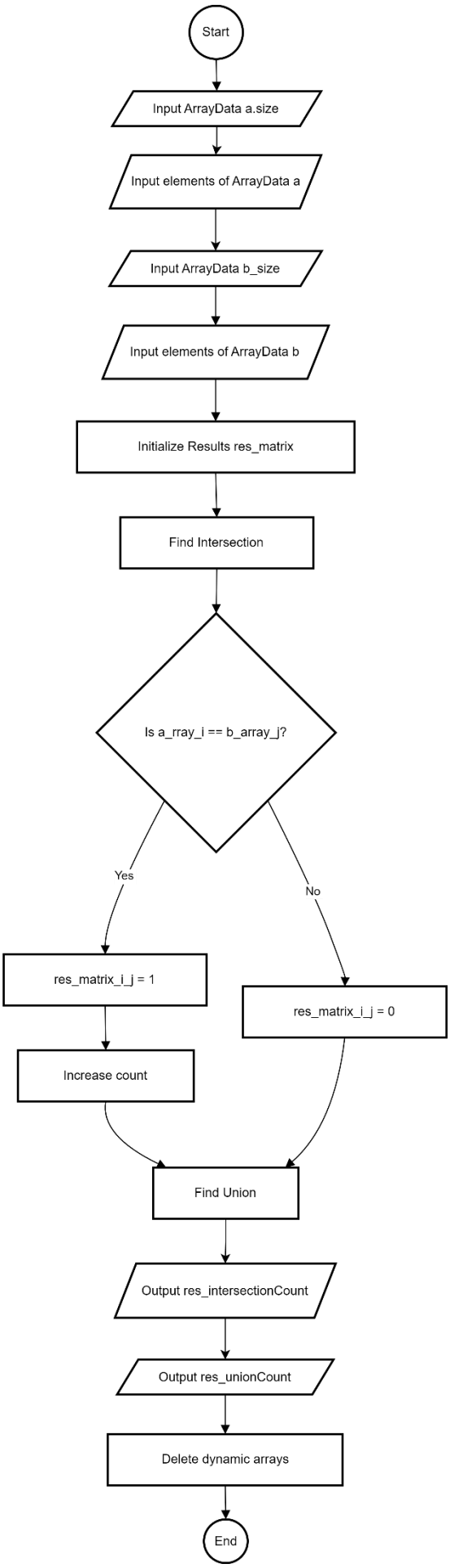
Завдання №6 - Self Practice Work – Algotester Lab 3 – Variant 2.

practice_work_self_algotester_tasks_yuliia_lazarevych.cpp > ...

```cpp
#include <iostream>
#include <set>

using namespace std;

//використовую структуру для зберігання масивів і їх розміру
struct ArrayData {
    int* array; //використовую динамічний масив
    int size;
};

//використовую вкладену структуру для зберігання результатів обробки
struct Results {
    int** matrix;     //використовую матрицю для порівняння (ДВОМІРНИЙ МАСИВ)
    int intersectionCount;
    int unionCount;
};

//використовую посилання для передачі структури та уникнення копіювання
int findIntersection(const ArrayData& a, const ArrayData& b, Results& res) {
    int count = 0;
    for (int i = 0; i < a.size; ++i) {
        for (int j = 0; j < b.size; ++j) {
            if (a.array[i] == b.array[j]) {
                res.matrix[i][j] = 1;
                ++count;
            } else {
                res.matrix[i][j] = 0;
            }
        }
    }
    return count;
}

//використовую множини для зберігання унікальних елементів (СТРУКТУРИ ДАНИХ)
int findUnion(const ArrayData& a, const ArrayData& b) {
    set<int> uniqueElements; //використовую множину для автоматичного видалення повторень
    for (int i = 0; i < a.size; ++i) {
        uniqueElements.insert(a.array[i]);
    }
    for (int i = 0; i < b.size; ++i) {
        uniqueElements.insert(b.array[i]);
    }
    return uniqueElements.size();
}
```

```cpp
int main() {
    ArrayData a; //використовую структуру для зберігання масиву
    cin >> a.size;
    a.array = new int[a.size]; //використовую динамічний масив для зберігання елементів
    for (int i = 0; i < a.size; ++i) {
        cin >> a.array[i];
    }

    ArrayData b; //використовую структуру для зберігання масиву
    cin >> b.size;
    b.array = new int[b.size]; //використовую динамічний масив для зберігання елементів
    for (int i = 0; i < b.size; ++i) {
        cin >> b.array[i];
    }

    Results res; //використовую вкладену структуру
    res.matrix = new int*[a.size]; //використовую динамічний двомірниц масив
        for (int i = 0; i < a.size; ++i) {
        res.matrix[i] = new int[b.size];
    }

    res.intersectionCount = findIntersection(a, b, res); //використовую алгоритм обробки при пошуку спільних елементів
    res.unionCount = findUnion(a, b); //використовую алгоритм обробки при пошуку унікальних елементів

    cout << res.intersectionCount << endl;
    cout << res.unionCount << endl;

    delete[] a.array;
    delete[] b.array;
    for (int i = 0; i < a.size; ++i) {
        delete[] res.matrix[i];
    }
    delete[] res.matrix;

    return 0;
}
```

Створено	Компілятор	Результат	Час (сек.)	Пам'ять (МіБ)	Дії
декілька секунд тому	C++ 23	Зараховано	0.003	1.188	Перегляд


Start
Input ArrayData a.size
Input elements of ArrayData a
Input ArrayData b_size
Input elements of ArrayData b
Initialize Results res_matrix
Find Intersection
Is a_rray_i == b_array_j?
Yes
No
res_matrix_i_j = 1
res_matrix_i_j = 0
Increase count
Find Union
Output res_intersectionCount
Output res_unionCount
Delete dynamic arrays
End

Додаткове завдання №7 – Algotester Lab 3 – Variant 1.

bonus_task_algotester.cpp > ...

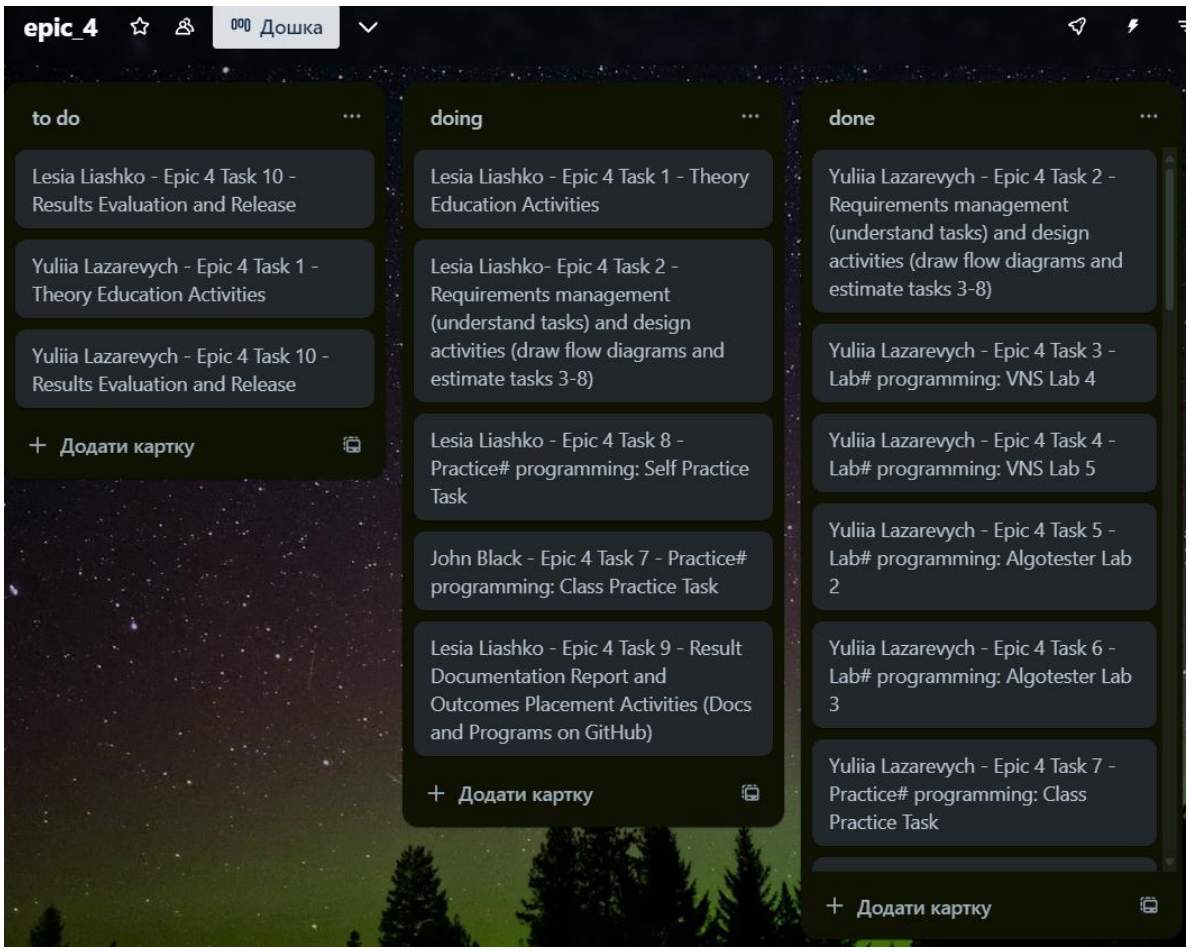
```cpp
#include <iostream>
#include <vector>
#include <string>

using namespace std;

//використовую функцію для підрахунку мінімальної кількості ритуалів
int countHealingRituals(vector<string>& lines) {
    int rituals = 0;

    for (const string& line : lines) {
        bool inDamagedWord = false;
        for (char c : line) {
            if (c == '#') {
                if (!inDamagedWord) {
                    inDamagedWord = true;
                    rituals++;
                }
            } else {
                inDamagedWord = false;
            }
        }
    }
    return rituals;
}

int main() {
    int N;
    cin >> N;
    cin.ignore();
    //використовую динамічний масив для зберігання рядків закляття
    vector<string> lines(N);

    for (int i = 0; i < N; i++) {
        getline(cin, lines[i]);
    }

    int result = countHealingRituals(lines);

    cout << result << endl;

    return 0;
}
```

Створено	Компілятор	Результат	Час (сек.)	Пам'ять (МіБ)	Дії
декілька секунд тому	C++ 23	Зараховано	0.017	1.297	Перегляд

Showing 1 to 1 of 1 rows

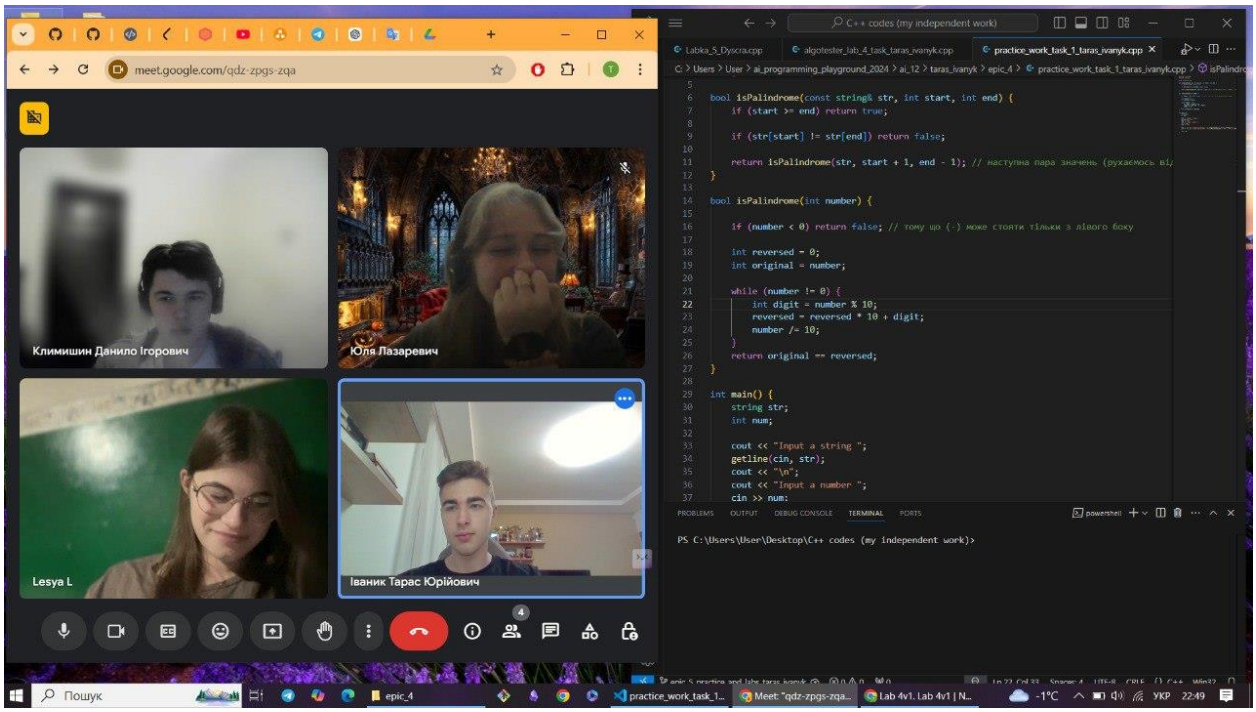
3. Кооперація з командою:

- Скрін прогресу по Трелло



Відстежували прогрес всієї команди завдяки дошці Trello

- Скрін з 2-ї зустрічі по обговоренню задач Епіку та Скрін прогресу по Трелло



Зустрічалися багато разів для обговорення та спільного виконання епіків.

Висновки: Виконуючи цей епік я ознайомилась з одновимірними, двовимірними та динамічними масивами, вказівниками та посиланнями, структурами даних, вкладеними структурами та алгоритмами обробки. Також я написала шість кодів де застосувала нові знання на практиці.