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

Осінна Єлизавета Сергіївна

# 

# **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:**

# Одержання навичок обробки одновимірних масивів. Організувати обробку масивів з використанням функцій, навчитися передавати масиви як параметри функцій.

# **Теоретичні відомості:**

1. Класи пам'яті у C++

* Статична пам’ять.
* Динамічна пам’ять.
* Поняття стеку.
* Виділення та вивільнення пам’яті.

1. Вступ до Масивів і Вказівників:

○      Основи масивів: визначення, важливість, приклади використання.

○      Різниця між статичними та динамічними масивами.

○      Основи вказівників: що це таке, як вони працюють.

○      Взаємозв'язок між масивами та вказівниками.

○      Вступ до посилань: основні концепції та відмінності від вказівників.

1. Одновимірні Масиви:

○      Створення та ініціалізація одновимірних масивів.

○      Основні операції: індексація, присвоєння, читання.

○      Цикли та обхід масивів.

○      Використання функцій для роботи з масивами.

○      Приклади алгоритмів сортування та пошуку.

1. Вказівники та Посилання:

○      Використання вказівників для доступу до елементів масиву.

○      Арифметика вказівників.

○      Різниця між вказівниками та посиланнями в контексті функцій.

○      Динамічне виділення пам'яті з використанням вказівників.

○      Використання вказівників для створення складних структур даних.

1. Двовимірні Масиви:

○      Оголошення та ініціалізація двовимірних масивів.

○      Вкладені цикли для обходу двовимірних масивів.

○      Практичні приклади використання двовимірних масивів.

○      Передача двовимірних масивів у функції.

○      Застосування двовимірних масивів для розв'язання задач.

1. Динамічні Масиви:

○      Основи динамічного виділення пам'яті.

○      Створення та управління динамічними масивами.

○      Використання операторів new та delete для управління пам'яттю.

○      Реалізація змінної розмірності масивів.

○      Передача динамічних масивів у функції.

1. Структури Даних:

○      Оголошення та використання структур.

○      Використання масивів та вказівників у структурах.

○      Функції для обробки даних у структурах.

○      Використання структур для представлення складних даних.

○      Вкладені структури та їх використання.

○  Об’єднання (Union)

○  Переліки (enumerations)

1. Вкладені Структури:

○      Поняття вкладених структур та їх оголошення.

○      Взаємодія з вкладеними структурами.

○      Використання вкладених структур для моделювання складних даних.

○      Передача вкладених структур у функції.

○      Приклади реального використання вкладених структур.

9.     Використання структур

-        Перевантаження операторів у структурі.

-        Вивід/ввід структури (operator<<);

-        Арифметичні операції з структурами (operator+, operator-);

-        Практичні задачі на виведення структур та операції з ними

1. Алгоритми обробки та робота з масивами та структурами:

○      Алгоритми пошуку та сортування в масивах.

○      Обробка та маніпуляції з даними у структурах.

○      Використання циклів та умовних операторів для роботи з масивами та структурами.

○      Інтеграція масивів та структур у алгоритми.

○      Розв'язання практичних задач з використанням масивів та структур.

1. Індивідуальний план опрацювання теорії:

* [C++ ⦁ Теорія ⦁ Урок 40 ⦁ Одновимірні масиви](https://www.youtube.com/watch?v=ULdbOaMBPYc)
* [Структури в С++ / Уроки по С++ / aCode](https://acode.com.ua/urok-64-struktury/)
* [Масиви в С++ / Уроки по С++ / aCode](https://acode.com.ua/urok-77-masyvy/)
* [Двовимірні масиви в С++ / aCode](https://acode.com.ua/urok-81-bagatovymirni-masyvy/)
* [Динамічне виділення пам'яті в С++ / Уроки по С++ / aCode](https://acode.com.ua/urok-89-dynamichne-vydilennya-pam-yati/)

# **Виконання роботи:**

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

**Завдання № 1 VNS Lab 4**

* Варіант 5
* Деталі завдання:

1) Сформувати одновимірний масив цілих чисел, використовуючи генератор

випадкових чисел.

2) Роздрукувати отриманий масив.

3) Знищити елементи кратні 7.

4) Додати після кожного непарного елемента масиву елемент зі значенням 0.

5) Роздрукувати отриманий масив.

**Завдання № 2 VNS Lab 5**

* Варіант 5
* Деталі завдання^

Написати функцію для знищення рядка із двовимірного масиву. Рядки, що

залишились, повинні бути розташовані щільно, елементи яких бракує

замінюються 0. За допомогою розроблених функцій знищити з масиву рядки

з номерами від А до В.

**Завдання № 3 Algotester Lab 2**

* Варіант 2
* Деталі завдання:

У вас є масив r розмiром N. Також вам дано 3 цiлих числа.

Спочатку ви маєте видалити з масиву цi 3 числа, якi вам данi. Пiсля цього перетворити цей

масив у масив сум, розмiром Nnew − 1 (розмiр нового масиву пiсля видалення елементiв), який

буде вiдображати суми сусiднiх елементiв нового масиву.

Далi необхiдно вивести масив сум на екран.

Вхiднi данi

У першому рядку цiле число N - кiлькiсть чисел

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

У третьому рядку 3 цiлих числа, a, b, c, якi треба видалити з масиву

Вихiднi данi

У першому рядку цiле число M - кiлькiсть чисел у масивi, який буде виведено

У наступному рядку M чисел - новий масив

Обмеження

1 ≤ N ≤ 105

0 ≤ ri ≤ 105

0 ≤ a, b, c ≤ 105

**Завдання № 4 Algotester Lab 3**

* Варіант 2
* Деталі завдання:

Вам дано 2 масиви розмiром N та M. Значення у цих масивах унiкальнi.

Ваше завдання вивести у першому рядку кiлькiсть елементiв, якi наявнi в обох масивах одно-

часно, у другому кiлькiсть унiкальних елементiв в обох масивах разом.

Вхiднi данi

У першому рядку цiле число N

у другому рядку N цiлих чисел a1..an

У третьому рядку цiле число M

у четвертому рядку M цiлих чисел b1..bn

Вихiднi данi

У першому рялку одне цiле число - кiлькiсть елементiв, якi наявнi в обох масивах одночасно.

У другому рядку кiлькiсть унiкальних елементiв в обох масивах (тобто кiлькiсть унiкальних

елементiв у масивi, який буде об’єднанням двох даних).

Обмеження

0 ≤ N ≤ 100

0 ≤ ai ≤ 100

0 ≤ M ≤ 100

0 ≤ bi ≤ 100

**Завдання № 5 Class Practice Work**

* Деталі завдання

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**

* Деталі завдання

# Lab 3v3

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 Scompressed

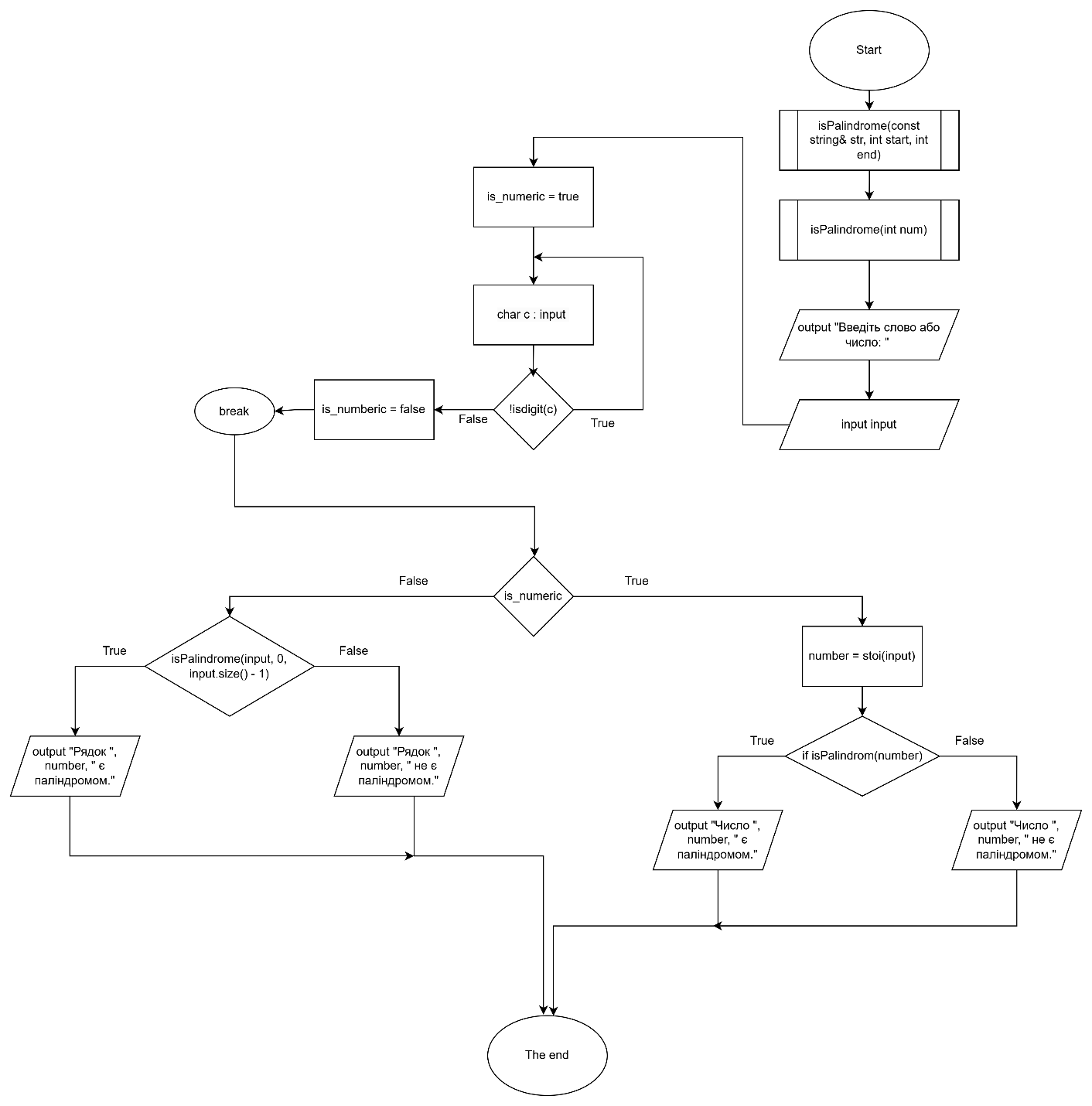
# Constraints

1≤|S|≤105

## **2. Дизайн та планована оцінка часу виконання завдань:**

Програма № 5 Class Practice Work

* Блок-схема



* Планований час на реалізацію кожної програми: 45 хв

## **4. Код програм з посиланням на зовнішні ресурси:**

Завдання № 1 VNS Lab 4

#include <iostream>

#include <cstdlib>

#include <ctime>

using namespace std;

int main() {

    // Ініціалізуємо генератор випадкових чисел

    srand(time(0));

    // 1. Створення одновимірного динамічного масиву цілих чисел

    const int SIZE = 20; // Початковий розмір масиву

    int\* array = new int[SIZE];

    int current\_size = SIZE;

    // Заповнення масиву випадковими числами від 0 до 99

    for (int i = 0; i < current\_size; ++i) {

        array[i] = rand() % 100;

    }

    // 2. Роздрукувати отриманий масив

    cout << "Початковий масив:" << endl;

    for (int i = 0; i < current\_size; ++i) {

        cout << array[i] << " ";

    }

    cout << endl;

    // 3. Видалення елементів, кратних 7

    // Підрахунок кількості елементів, які залишаться

    int new\_size = 0;

    for (int i = 0; i < current\_size; ++i) {

        if (array[i] % 7 != 0) {

            ++new\_size;

        }

    }

    // Створюємо новий масив для збереження елементів, які не є кратними 7

    int\* filtered\_array = new int[new\_size];

    int index = 0;

    for (int i = 0; i < current\_size; ++i) {

        if (array[i] % 7 != 0) {

            filtered\_array[index++] = array[i];

        }

    }

    // Оновлення розміру і заміна масиву

    delete[] array;

    array = filtered\_array;

    current\_size = new\_size;

    // 4. Додавання 0 після кожного непарного елемента

    // Підрахунок нового розміру масиву з урахуванням вставок

    new\_size = current\_size;

    for (int i = 0; i < current\_size; ++i) {

        if (array[i] % 2 != 0) {

            ++new\_size; // Кожен непарний елемент потребує додаткового місця для 0

        }

    }

    // Створення нового масиву для зберігання результату

    int\* modified\_array = new int[new\_size];

    index = 0;

    for (int i = 0; i < current\_size; ++i) {

        modified\_array[index++] = array[i];

        if (array[i] % 2 != 0) {

            modified\_array[index++] = 0; // Додаємо 0 після непарного елемента

        }

    }

    // Оновлення розміру і заміна масиву

    delete[] array;

    array = modified\_array;

    current\_size = new\_size;

    // 5. Роздрукувати отриманий масив

    cout << "Змінений масив:" << endl;

    for (int i = 0; i < current\_size; ++i) {

        cout << array[i] << " ";

    }

    cout << endl;

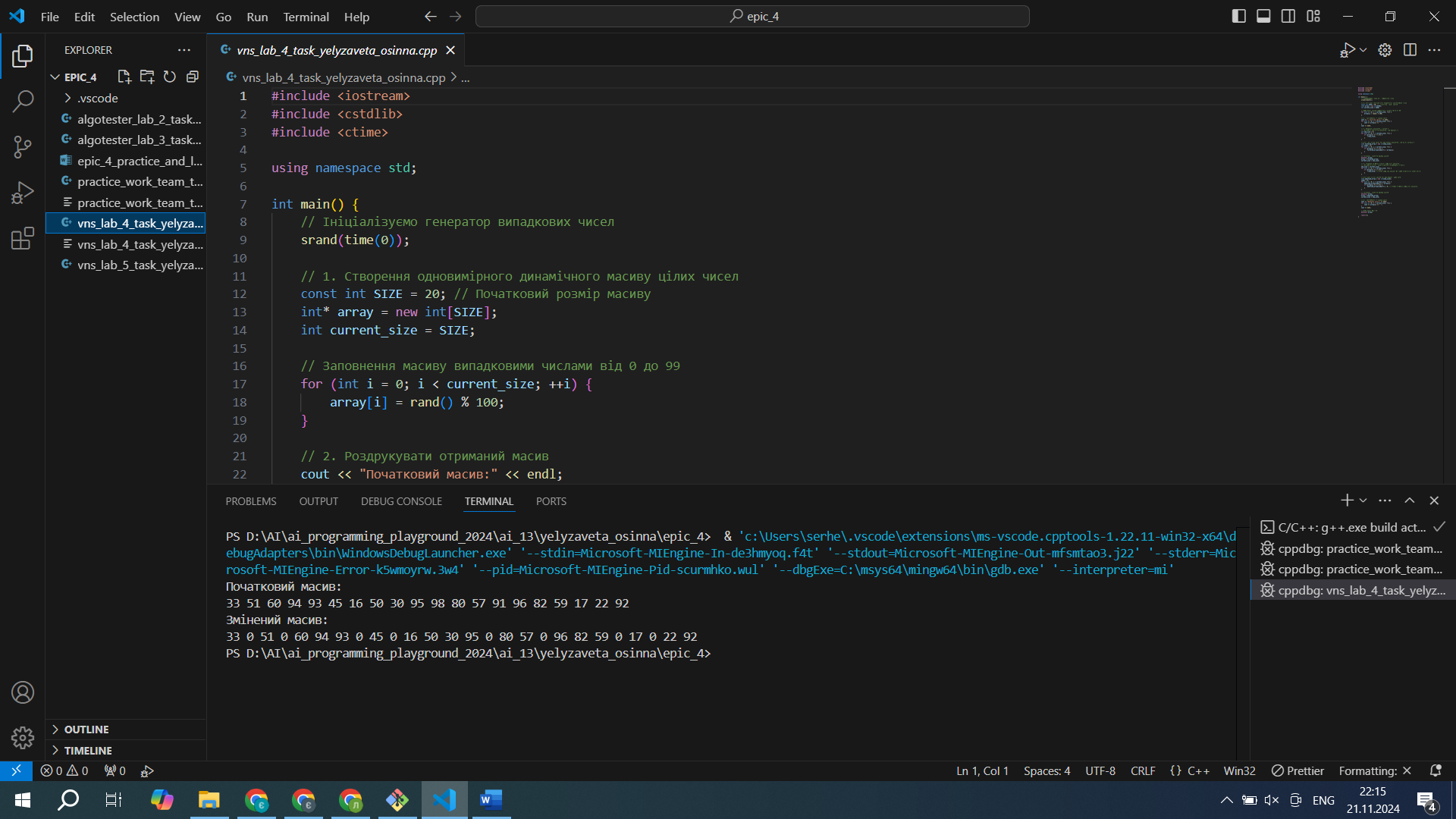
    // Звільнення пам'яті

    delete[] array;

    return 0;

}

Результат:



Час затрачений на виконання завдання: 1 год

Завдання № 2 VNS Lab 5

#include <iostream>

using namespace std;

// Функція для видалення рядків з індексів A до B

void deleteRows(int\*\* matrix, int& rows, int cols, int A, int B) {

    int shift = B - A + 1; // Кількість рядків для видалення

    // Зсуваємо рядки, які залишились, вгору

    for (int i = A; i + shift < rows; ++i) {

        for (int j = 0; j < cols; ++j) {

            matrix[i][j] = matrix[i + shift][j];

        }

    }

    // Заповнюємо останні рядки нулями

    for (int i = rows - shift; i < rows; ++i) {

        for (int j = 0; j < cols; ++j) {

            matrix[i][j] = 0;

        }

    }

    // Оновлюємо кількість рядків

    rows -= shift;

}

int main() {

    int rows = 6, cols = 5; // Початковий розмір матриці

    int A = 1, B = 3;       // Індекси рядків для видалення

    // Створення двовимірного динамічного масиву

    int\*\* matrix = new int\*[rows];

    for (int i = 0; i < rows; ++i) {

        matrix[i] = new int[cols];

    }

    // Ініціалізація матриці

    int counter = 1;

    for (int i = 0; i < rows; ++i) {

        for (int j = 0; j < cols; ++j) {

            matrix[i][j] = counter++;

        }

    }

    // Вивід початкової матриці

    cout << "Початкова матриця:" << endl;

    for (int i = 0; i < rows; ++i) {

        for (int j = 0; j < cols; ++j) {

            cout << matrix[i][j] << " ";

        }

        cout << endl;

    }

    // Видалення рядків з A до B

    deleteRows(matrix, rows, cols, A, B);

    // Вивід оновленої матриці

    cout << "\nОновлена матриця після видалення рядків:" << endl;

    for (int i = 0; i < rows; ++i) {

        for (int j = 0; j < cols; ++j) {

            cout << matrix[i][j] << " ";

        }

        cout << endl;

    }

    // Видалення динамічної пам'яті

    for (int i = 0; i < rows + (B - A + 1); ++i) {

        delete[] matrix[i];

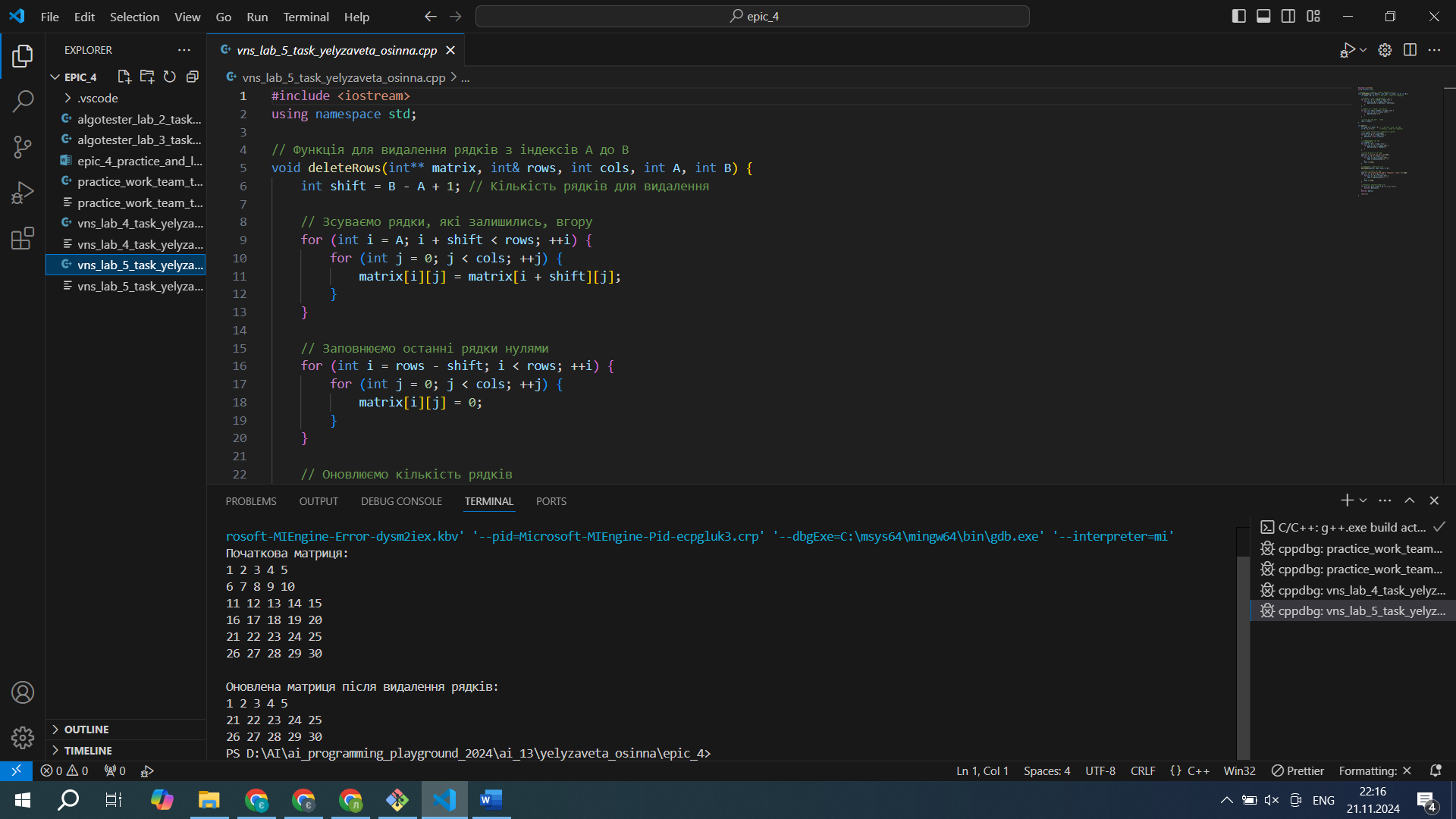
    }

    delete[] matrix;

    return 0;

}

Результат:



Час затрачений на виконання завдання: 1 год

Завдання № 3 Algotester Lab 2

#include <iostream>

using namespace std;

int main() {

    int N;

    cin >> N;

    int\* r = new int[N];

    for (int i = 0; i < N; ++i) {

        cin >> r[i];

    }

    int a, b, c;

    cin >> a >> b >> c;

    // Масив для фільтрованих елементів (максимальний розмір такий самий, як у `r`)

    int\* filtered = new int[N];

    int new\_size = 0;

    // Фільтрація елементів без використання set

    for (int i = 0; i < N; ++i) {

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

            filtered[new\_size++] = r[i];

        }

    }

    // Масив для сум сусідніх елементів

    if (new\_size > 1) {

        int\* sums = new int[new\_size - 1];

        for (int i = 0; i < new\_size - 1; ++i) {

            sums[i] = filtered[i] + filtered[i + 1];

        }

        // Вивід результатів

        cout << (new\_size - 1) << endl;

        for (int i = 0; i < new\_size - 1; ++i) {

            cout << sums[i] << " ";

        }

        cout << endl;

        delete[] sums;  // Звільняємо пам'ять для масиву сум

    } else {

        // Якщо не вистачає елементів для створення пар

        cout << 0 << endl;

    }

    // Звільнення пам'яті

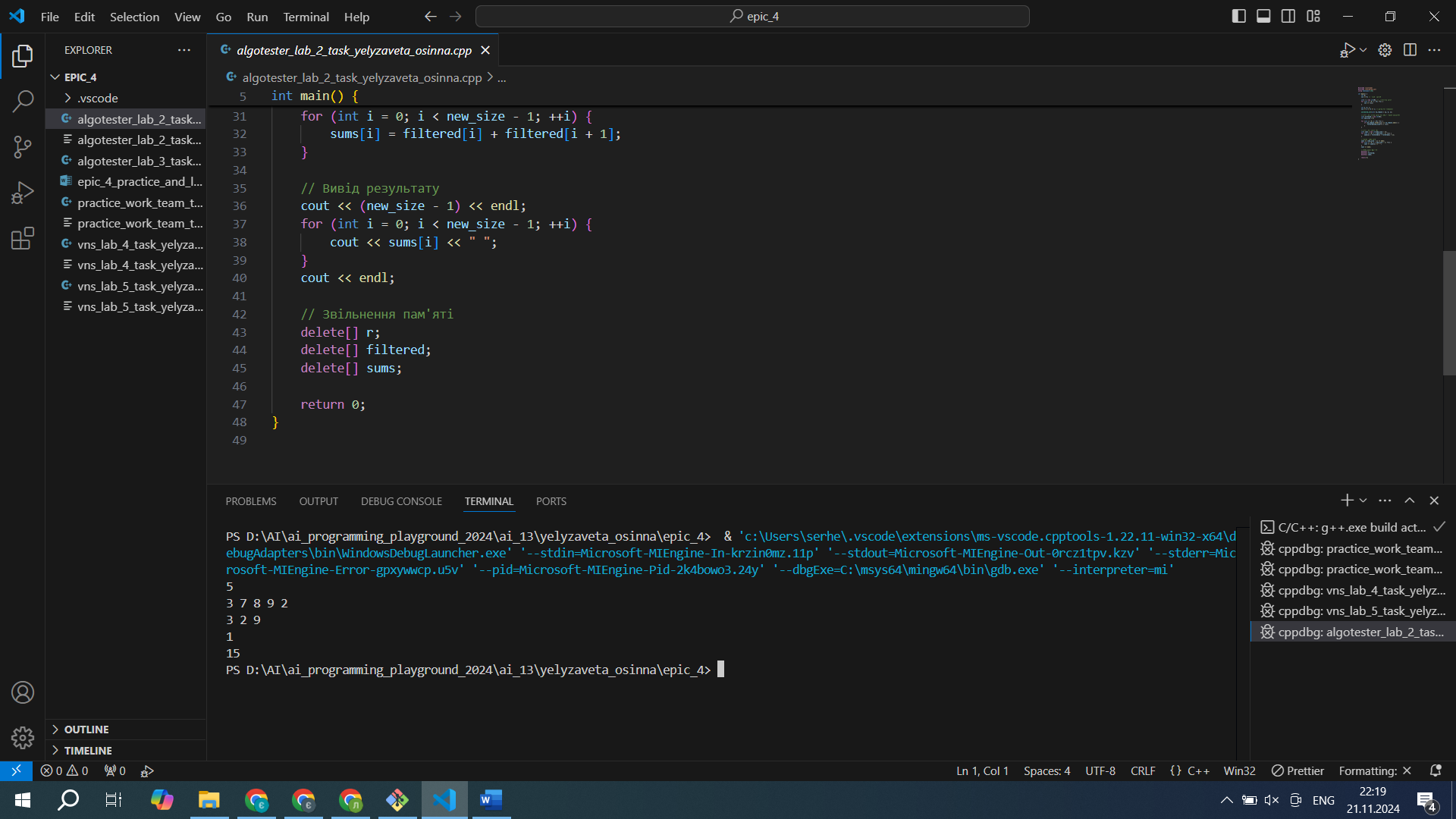
    delete[] r;

    delete[] filtered;

    return 0;

}

Результат:



Час затрачений на виконання завдання: 1 год

Завдання № 4 Algotester Lab 3

#include <iostream>

using namespace std;

int main() {

    int N, M;

    // Зчитування масиву a

    cin >> N;

    int\* a = new int[N];

    for (int i = 0; i < N; i++) {

        cin >> a[i];

    }

    // Зчитування масиву b

    cin >> M;

    int\* b = new int[M];

    for (int i = 0; i < M; i++) {

        cin >> b[i];

    }

    // Знаходимо кількість спільних елементів

    int common\_elements = 0;

    for (int i = 0; i < N; i++) {

        for (int j = 0; j < M; j++) {

            if (a[i] == b[j]) {

                common\_elements++;

                break; // Переходимо до наступного елемента з a

            }

        }

    }

    // Знаходимо кількість унікальних елементів

    int\* union\_array = new int[N + M];

    int unique\_count = 0;

    // Додаємо елементи з a

    for (int i = 0; i < N; i++) {

        bool is\_unique = true;

        for (int j = 0; j < unique\_count; j++) {

            if (union\_array[j] == a[i]) {

                is\_unique = false;

                break;

            }

        }

        if (is\_unique) {

            union\_array[unique\_count++] = a[i];

        }

    }

    // Додаємо елементи з b

    for (int i = 0; i < M; i++) {

        bool is\_unique = true;

        for (int j = 0; j < unique\_count; j++) {

            if (union\_array[j] == b[i]) {

                is\_unique = false;

                break;

            }

        }

        if (is\_unique) {

            union\_array[unique\_count++] = b[i];

        }

    }

    // Вивід результатів

    cout << common\_elements << endl;

    cout << unique\_count << endl;

    // Звільнення пам'яті

    delete[] a;

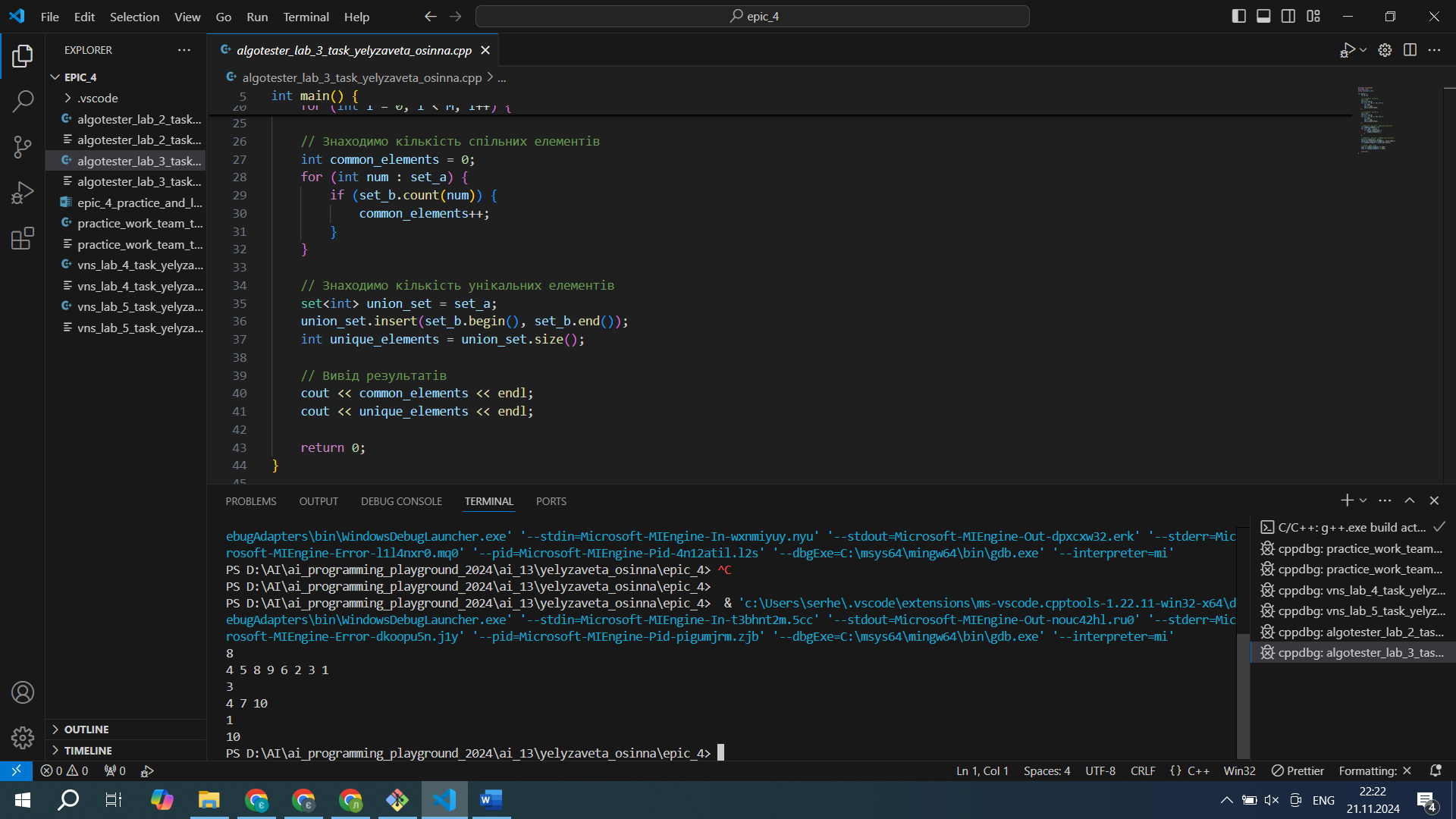
    delete[] b;

    delete[] union\_array;

    return 0;

}

Результат:



Час затрачений на виконання завдання: 1 год

Завдання № 5 Class Practice Work

#include <iostream>

#include <string>

using namespace std;

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 num) {

    if (num < 0) return false;

    int reversed = 0, original = num;

    while (num > 0) {

        reversed = reversed \* 10 + num % 10;

        num /= 10;

    }

    return original == reversed;

}

int main() {

    cout << "Введіть слово або число: ";

    string input;

    cin >> input;

    bool is\_numeric = true;

    for (char c : input) {

        if (!isdigit(c)) {

            is\_numeric = false;

            break;

        }

    }

    if (is\_numeric) {

        int number = stoi(input);

        if (isPalindrome(number)) {

            cout << "Число " << number << " є паліндромом." << endl;

        } else {

            cout << "Число " << number << " не є паліндромом." << endl;

        }

    } else {

        if (isPalindrome(input, 0, input.size() - 1)) {

            cout << "Рядок \"" << input << "\" є паліндромом." << endl;

        } else {

            cout << "Рядок \"" << input << "\" не є паліндромом." << endl;

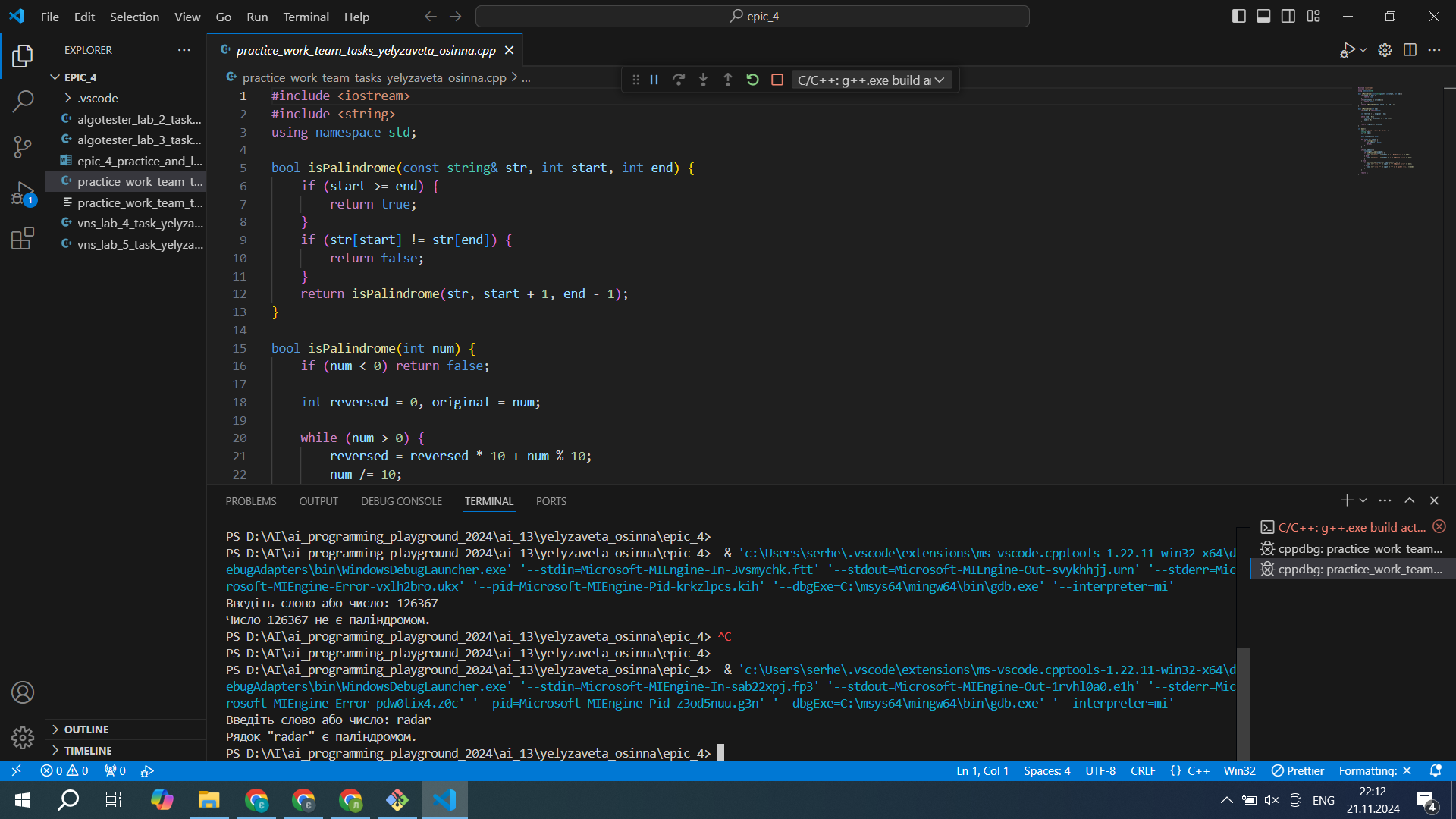
        }

    }

    return 0;

}

Результат:



Час затрачений на виконання завдання: 1 год

Завдання № 6 Self Practice Work

#include <iostream>

#include <cstring>

using namespace std;

void compressString(const char\* s, char\* compressed) {

    int n = strlen(s);

    int pos = 0;

    for (int i = 0; i < n; ++i) {

        char currentChar = s[i];

        int count = 1;

        while (i + 1 < n && s[i + 1] == currentChar) {

            ++count;

            ++i;

        }

        compressed[pos++] = currentChar;

        if (count > 1) {

            char countStr[10];

            sprintf(countStr, "%d", count);

            for (int j = 0; countStr[j] != '\0'; ++j) {

                compressed[pos++] = countStr[j];

            }

        }

    }

    compressed[pos] = '\0';

}

int main() {

    char s[100001];

    cin >> s;

    char compressed[200001];

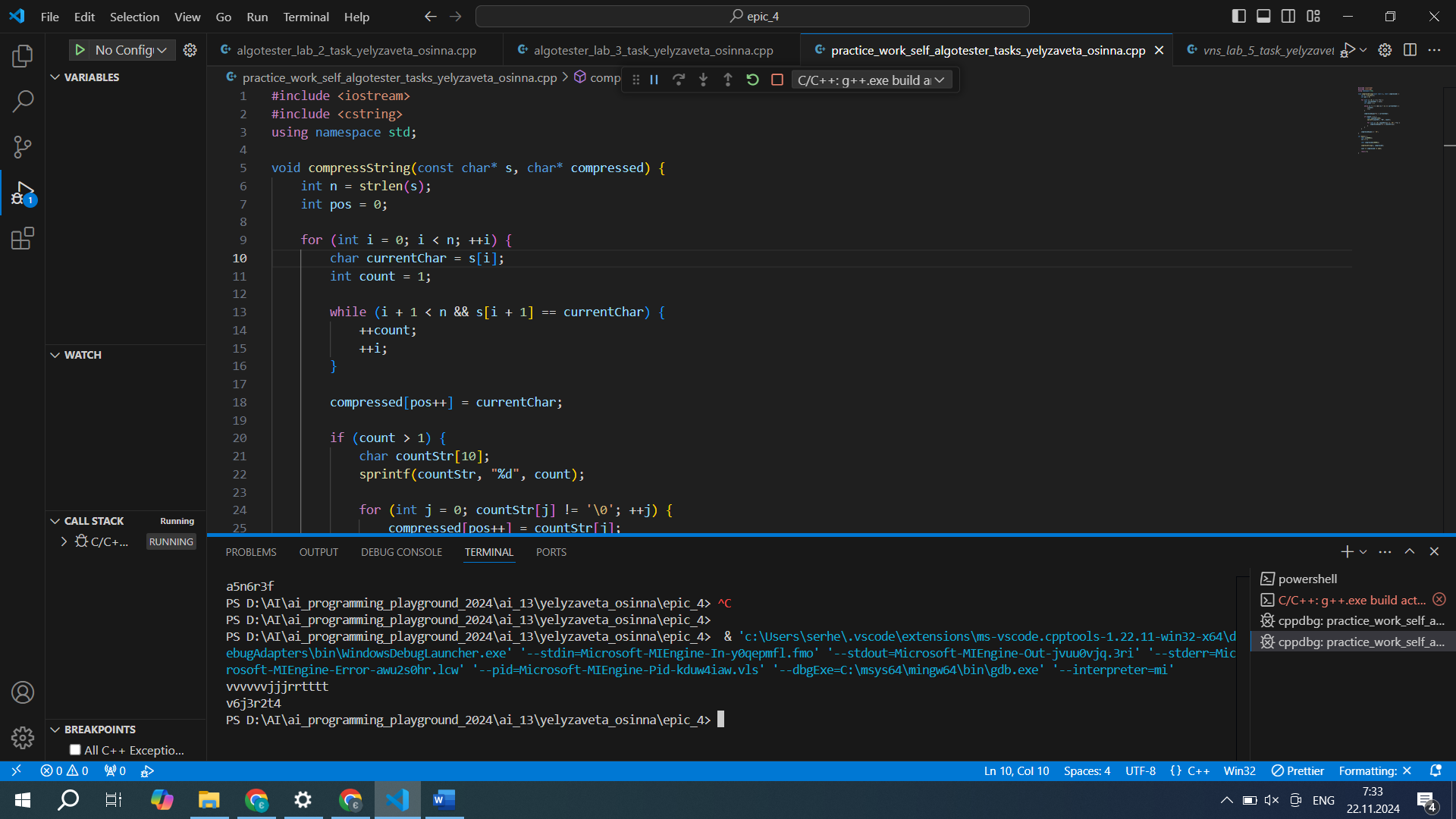
    compressString(s, compressed);

    cout << compressed << endl;

    return 0;

}

Результат:

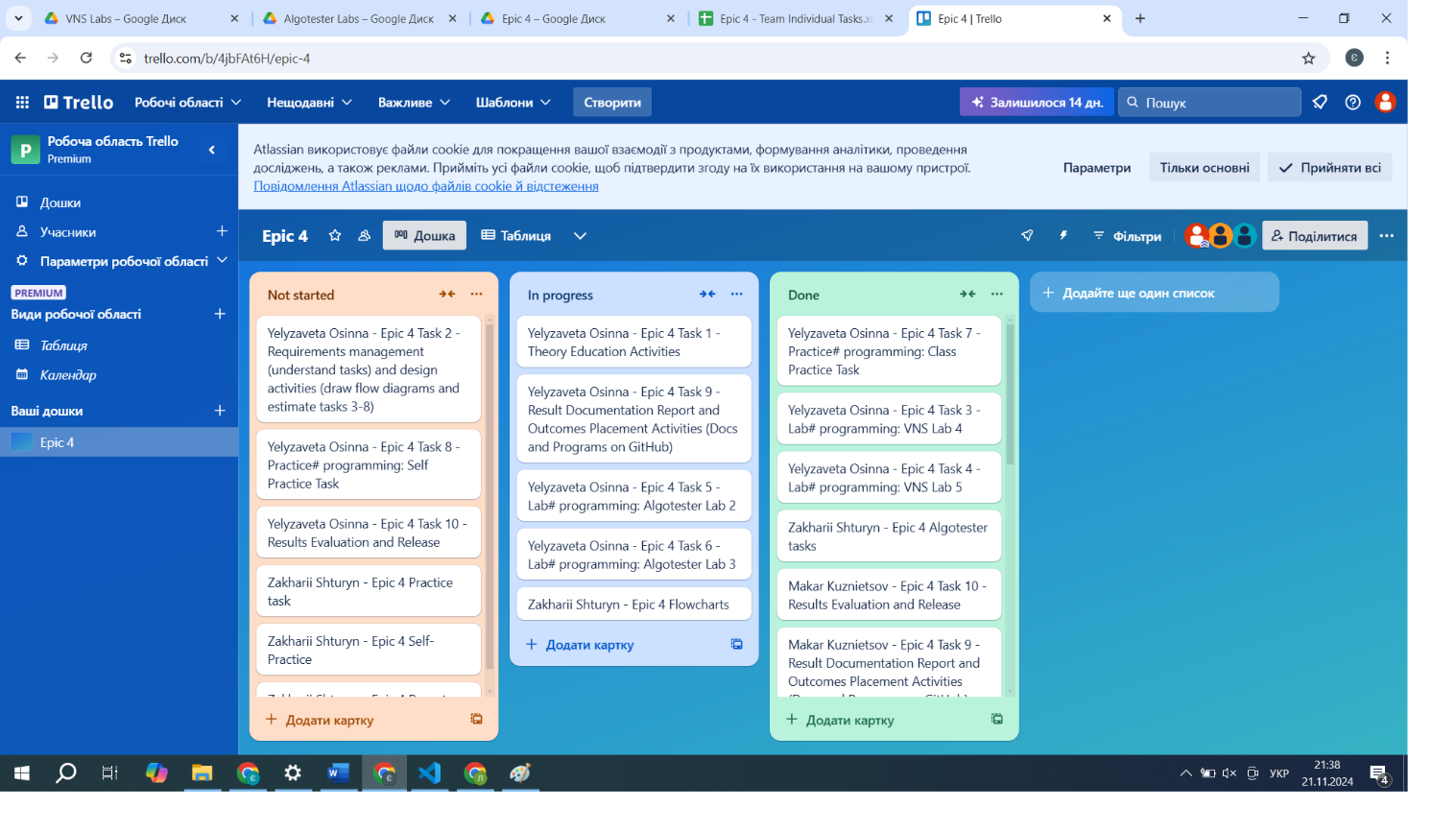


Час затрачений на виконання завдання: 1 год

## **6. Кооперація з командою:**

# 

Trello:



# **Висновки:**

# Виконуючи цю роботу я одержала навички обробки одновимірних масивів, обробки масивів з використанням функцій, навчилася передавати масиви як параметри функцій.