Міністерство освіти і науки України

Національний університет «Львівська політехніка»

Кафедра систем штучного інтелекту

A blue and white logo

Description automatically generated

**Звіт**

**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

**Виконав:**

Студент групи ШІ-11

Климчук Юрій Олегович

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:** Навчитися ефективно працювати з одновимірними та двовимірними масивами, використовувати вказівники та посилання для оптимізації роботи з динамічними масивами, розібратися зі структурами даних і вкладеними структурами для зберігання складних даних, а також освоїти алгоритми обробки та маніпуляції масивами та структурами.

**Теоретичні відомості:**

**1)Перелік тем:**

1. Класи пам'яті у C++
2. Вступ до Масивів і Вказівників
3. Одновимірні Масиви
4. Вказівники та Посилання
5. Двовимірні Масиви
6. Динамічні Масиви
7. Структури Даних
8. Вкладені Структури

9.     Використання структур

1. Алгоритми обробки та робота з масивами та структурами

**2)Індивідуальний план опрацювання теорії:**

1. **Класи пам'яті у C++**
   * [C++ Storage Classes](https://www.tutorialspoint.com/cplusplus/cpp_storage_classes.htm)
   * [Memory Classes in C++](https://www.geeksforgeeks.org/storage-classes-in-c/)
2. **Вступ до Масивів і Вказівників**
   * [Introduction to Arrays in C++](https://www.w3schools.com/cpp/cpp_arrays.asp)
   * [Pointers in C++](https://www.cplusplus.com/doc/tutorial/pointers/)
3. **Одновимірні Масиви**
   * [C++ One-Dimensional Arrays](https://www.w3schools.com/cpp/cpp_arrays.asp)
   * [Array Basics in C++](https://www.geeksforgeeks.org/arrays-in-c-cpp/)
4. **Вказівники та Посилання**
   * [Pointers and References in C++](https://www.cplusplus.com/doc/tutorial/pointers/)
   * [References in C++](https://www.geeksforgeeks.org/references-in-c/)
5. **Двовимірні Масиви**
   * [Two-Dimensional Arrays in C++](https://www.tutorialspoint.com/cplusplus/cpp_multi_dimensional_arrays.htm)
   * [2D Arrays in C++](https://www.geeksforgeeks.org/multidimensional-arrays-c-cpp/)
6. **Динамічні Масиви**
   * [Dynamic Arrays in C++](https://www.geeksforgeeks.org/dynamic-arrays-in-c-cpp/)
   * [C++ Dynamic Memory](https://www.tutorialspoint.com/cplusplus/cpp_dynamic_memory.htm)
7. **Структури Даних**
   * [Structures in C++](https://www.w3schools.com/cpp/cpp_structs.asp)
   * [Introduction to Data Structures](https://www.geeksforgeeks.org/data-structures/)
8. **Вкладені Структури**
   * [Nested Structures in C++](https://www.geeksforgeeks.org/nested-structures-in-c/)
   * [C++ Nested Structures](https://www.tutorialspoint.com/structure-within-structure-in-cplusplus)
9. **Використання структур**
   * [C++ Structs and their Use](https://www.w3schools.com/cpp/cpp_structs.asp)
   * [Uses of Structures in C++](https://www.geeksforgeeks.org/structures-in-cpp/)
10. **Алгоритми обробки та робота з масивами та структурами**
    * [Algorithms for Arrays in C++](https://www.geeksforgeeks.org/array-data-structure/)
    * [Working with Arrays and Structures in C++](https://www.tutorialspoint.com/cplusplus/cpp_structures.htm)

**Виконання роботи:**

**1)Перелік завдань:**

* John Black - Epic 4 Task 1 - Theory Education Activities
* John Black - Epic 4 Task 2 - Requirements management (understand tasks) and design activities (draw flow diagrams and estimate tasks 3-8)
* John Black - Epic 4 Task 3 - Lab# programming: VNS Lab 4(варіант 4)
* John Black - Epic 4 Task 4 - Lab# programming: VNS Lab 5(варіант 4)
* John Black - Epic 4 Task 5 - Lab# programming: Algotester Lab 2(варіант 1)
* John Black - Epic 4 Task 6 - Lab# programming: Algotester Lab 3(варіант 3)
* John Black - Epic 4 Task 7 - Practice# programming: Class Practice Task
* John Black - Epic 4 Task 8  - Practice# programming:  Self Practice Task
* John Black - Epic 4 Task 9  - Result Documentation Report and Outcomes Placement Activities (Docs and Programs on GitHub)
* John Black - Epic 4 Task 10 - Results Evaluation and Release

**2)Умови завдань:**

**Task 3:**

1) Сформувати одновимірний масив цілих чисел, використовуючи генератор випадкових чисел. 2) Роздрукувати отриманий масив. 3) Знищити елементи, індекси яких кратні 3. 4) Додати після кожного від’ємного елемента масиву елемент зі значенням | M[ I-1 ]+1|. 5) Роздрукувати отриманий масив.

**Task4: Використовуючи функції, розв’язати зазначене у варіанті завдання. Масив повинен передаватися у функцію як параметр.**

Визначити чи є матриця ортонормованою, тобто такою, що скалярний добуток кожної пари різних рядків дорівнює 0, а скалярний добуток рядка самого на себе дорівнює 1.

**Task5:**

**Lab 2v1**

*Limits: 1 sec., 256 MiB*

У вас є дорога, яка виглядає як N чисел.

Після того як ви по ній пройдете - вашу втому можна визначити як різницю максимального та мінімального елементу.

Ви хочете мінімізувати втому, але все що ви можете зробити - викинути одне число з дороги, тобто забрати його з масиву.

В результаті цієї дії, яку мінімальну втому ви можете отримати в кінці дороги?

**Input**

У першому рядку ціле число N - кількість чисел

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

**Output**

Єдине ціле число mm - мінімальна втома, яку можна отримати

# **Task6:** **Lab 3v3**

*Limits: 1 sec., 256 MiB*

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

# **Task7: Перевірка чи слово або число є паліндромом**

# **Задача**

## **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:

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

1. Приклад визначення функції:

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

1. Перевантаження функцій:

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

b. *bool isPalindrome*(ціле число);

1. Рекурсія:

a.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.

**Task 8:**

**Lab 2v2**

*Обмеження: 1 сек., 256 МіБ*

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 Nnew−1 (розмір нового масиву після видалення елементів), який буде відображати суми сусідніх елементів нового масиву.

Далі необхідно вивести масив сум на екран.

**Вхідні дані**

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

**Вихідні дані**

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

У наступному рядку M чисел - новий масив

**3)Дизайн та планова оцінка часу виконання завдань:**

Task 4 - Lab# programming: VNS Lab 5(варіант 4)

Зображення, що містить текст, схема, План, Креслення

Автоматично згенерований опис

Орієнтовний час виконання: 50хв

**4)Код програм з посиланням на зовнішні ресурси:**

Task 3 - Lab# programming: VNS Lab 4(варіант 4)

Посилання на файл програми: <https://github.com/artificial-intelligence-department/ai_programming_playground_2024/pull/269/files#diff-2a2d971a5e4460f2f9fe4453ee9f28cc39fabc8dd8232d540a61245c7ee49c5b>

#include <iostream>

#include <stdio.h>

#include <algorithm>

#include <vector>

using namespace std;

int main(){

    //створюємо і виводимо наш вектор рандомних чисел

    vector<int> array = {30, -84, -19, 90, 68, -29, 73, 2, 3, -22, 86, -60, -23, -35, 56, 80, 83, 62, -48, 94};

    int size = array.size();

    cout << "Original array: " << endl;

    for(int i = 0; i < size; i++){

        if(i == (size-1)){

            cout << array[i] << "." << endl;

        }

        else{

            cout << array[i] << ", ";

        }

    }

    //створюємо новий вектор куди записуємо всі елементи індекси яких ділиться на 3

    vector<int> modified\_array = array;

    for(int i = 0; i < size; i++){

        if(i%3 == 0){

            int element = array.at(i);

            auto it = find(modified\_array.begin(), modified\_array.end(), element);

            modified\_array.erase(it);

        }

    }

    //вставляємо після від'ємних елементів нові числа

    int size\_1 = modified\_array.size();

    for(int i = 0; i < size\_1; i++){

        if(modified\_array[i] < 0 && i != 0){

            modified\_array.insert(modified\_array.begin()+(i+1), abs(modified\_array[i-1]+1));

        }

    }

    //виводимо наш кінцевий масив

    int size\_2 = modified\_array.size();

    cout << "Modified array: " << endl;

    for(int i = 0; i < size\_2; i++){

        if(i == (size\_2-1)){

            cout << modified\_array[i] << ".";

        }

        else{

            cout << modified\_array[i] << ", ";

        }

    }

    return 0;

}

Task 4 - Lab# programming: VNS Lab 5(варіант 4)

Посилання на файл програми: <https://github.com/artificial-intelligence-department/ai_programming_playground_2024/pull/269/files#diff-82b3649a70ca6d2a9eabcc467c9f01a0e3925570a79ff86849fe54c2f0fb2de0>

#include <iostream>

#include <cmath>

using namespace std;

//уводимо допустиму похибку для нецілих чисел

const double EPSILON = 1e-9;

//функція для перевірки на ортогональність

bool Orthonormal(double a[][3]){

    //перевіряємо чи добуток рядка самого на себе рівний 1

    int n = sizeof(a[0])/sizeof(a[0][0]);

    for(int i = 0; i < n; i++){

        double same\_multiply = 0.0;

        for(int j =0; j < n; j++){

            same\_multiply += a[j][i]\*a[j][i];

        }

        if ((same\_multiply - 1.0)> EPSILON){

            return false;

        }

    }

    //перевіряємо чи добуток рядків рівний 0

    for(int i = 0; i < n; i++){

        for(int j = i+1; j < n; j++){

            double different\_multiply = 0.0;

            for(int k = 0; k < n; k++){

                different\_multiply += a[k][i]\*a[k][j];

            }

            if (abs(different\_multiply)> EPSILON){

                return false;

            }

        }

    }

    return true;

}

int main(){

    //створюємо матрицю 3х3

    double matrix[3][3]{

        {1 / sqrt(2), 1 / sqrt(2), 0},

        {-1 / sqrt(2), 1 / sqrt(2), 0},

        {0, 0, 1}

    };

    //перевіряємо отримане значення з функції

    if(Orthonormal(matrix)){

        cout << "Matrix is orthonormal";

    }

    else{

        cout << "Matrix isn`t orthonormal";

    }

    return 0;

}

Task 5 - Lab# programming: Algotester Lab 2(варіант 1)

Посилання на файл програми: <https://github.com/artificial-intelligence-department/ai_programming_playground_2024/pull/269/files#diff-2157d7b677ff809686a21e35a12595f67da742445bbadcf8caec5e4e83c553be>

Посилання на алготестер: <https://algotester.com/uk/ContestProblem/DisplayWithEditor/135592>

#include <iostream>

#include <vector>

#include <algorithm>

using namespace std;

int main(){

    //розмір дороги

    int N;

    cin >> N;

    //сторюємо вектор і вводимо значення відстані

    vector<int> r;

    for(int i = 0; i < N; i++){

        int value;

        cin >> value;

        r.push\_back(value);

    }

    //знаходимо максимальний та мінімальний елементи

    auto minimal = min\_element(r.begin(), r.end());

    auto maximal = max\_element(r.begin(), r.end());

    //створюємо ветор у якому видаляємо максимальний елемент і знаходимо відстань

    vector <int> copy\_1 = r;

    auto it\_1 = find(copy\_1.begin(), copy\_1.end(), \*maximal);

    copy\_1.erase(it\_1);

    auto min\_1 = min\_element(copy\_1.begin(), copy\_1.end());

    auto max\_1 = max\_element(copy\_1.begin(), copy\_1.end());

    int difference\_1 = \*max\_1 - \*min\_1;

    //створюємо ветор у якому видаляємо мінімальний елемент і знаходимо відставнь

    vector <int> copy\_2 = r;

    auto it\_2 = find(copy\_2.begin(), copy\_2.end(), \*minimal);

    copy\_2.erase(it\_2);

    auto min\_2 = min\_element(copy\_2.begin(), copy\_2.end());

    auto max\_2 = max\_element(copy\_2.begin(), copy\_2.end());

    int difference\_2 = \*max\_2 - \*min\_2;

    //порівнюємо два елементи і виводимо менший

    cout << min(difference\_1, difference\_2);

    return 0;

}

Task 6 - Lab# programming: Algotester Lab 3(варіант 3)

Посилання на файл програми: <https://github.com/artificial-intelligence-department/ai_programming_playground_2024/pull/269/files#diff-47a2bc0f703d6113269be3f01df1362a1a86f1035137bc76f28b8871c0ba569f>

Посилання на алготестер: <https://algotester.com/uk/ContestProblem/DisplayWithEditor/135597>

#include <iostream>

#include <string>

using namespace std;

int main() {

    //створюмо змінну для рядка юзера

    string s;

    cin >> s;

    //створюмо змінні для стисненого строки та розміру

    string compressed;

    int n = s.size();

    for (int i = 0; i < n; i++) {

        char currentChar = s[i];

        int count = 1;

        //рахуємо кількість однвкових літер під ряд

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

            ++count;

            ++i;

        }

        //записуємо у нову строку літери і їх кількість

        compressed += currentChar;

        if (count > 1) {

            compressed += to\_string(count);

        }

    }

    //виводимо кінцевий результат

    cout << compressed << endl;

    return 0;

}

Task 7 - Practice# programming: Class Practice Task

Посилання на файл програми: <https://github.com/artificial-intelligence-department/ai_programming_playground_2024/pull/269/files#diff-90fed6a244ff448d05a1acc09cc58cae33b0d4cb8af47ab03431de16f6d42467>

#include <iostream>

#include <string>

using namespace std;

//оголошуємо перевантажену функцію для перевірки паліндрома

bool isPalindrome(string s, int start, int end);

bool isPalindrome(int n);

int main(){

    //створюємо строку для слова і перевіряємо на паліндром

    string line;

    cin >> line;

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

        cout << line <<" is a palindrome\n";

    }

    else {

        cout << line << " isn`t a palindrome\n";

    }

    //створюємо строку для числа і перевіряємо на паліндром

    int number;

    cin >> number;

    if (isPalindrome(number)){

        cout << number <<" is a palindrome";

    }

    else {

        cout << number << " isn`t a palindrome";

    }

    return 0;

}

bool isPalindrome(string s, int start, int end){

    //перевірка щоб змінні не зайшли за середину

    if(start <= end){

        //чи співпадають букви з двох кінців

        if(s[start] == s[end]){

            //створюмо рекурсію для перевірки всієї строки

            isPalindrome(s, ++start, ++end);

        }

        else{

            return false;

        }

    }

    return true;

}

bool isPalindrome(int n){

    //змінні для перевернутого і оригінального числа

    int reversed = 0;

    int original = n;

    while (n > 0) {

        //перевертаємо число через остачу ділення на 10

        int digit = n % 10;

        reversed = reversed \* 10 + digit;

        n /= 10;

    }

    //звіряємо оригінал і перевернуту строку

    return original == reversed;

}

Task 8  - Practice# programming:  Self Practice Task

Посилання на файл програми: <https://github.com/artificial-intelligence-department/ai_programming_playground_2024/pull/269/files#diff-e1e11e687589a6524c81ecb514ae49ecb7b624dba0f9c67dbc28b0124b36f740>

Посилання на алготестер: <https://algotester.com/uk/ContestProblem/DisplayWithEditor/135593>

#include <iostream>

#include <vector>

using namespace std;

int main() {

    //змінні для вводу користувача

    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;

    //записуємо не співпадаючі елементи у новий масив

    vector<int> r\_new;

    for (int i = 0; i < n; i++) {

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

            r\_new.push\_back(r[i]);

        }

    }

    //знаходимо розмір нового масиву і перевіряємо його

    int M = r\_new.size() - 1;

    if (M <= 0) {

        cout << 0;

        return 0;

    }

    //записуємо суму сусідних елементів у окремий масив

    vector<int> M\_sum(M);

    for (int i = 0; i < M; i++) {

        M\_sum[i] = r\_new[i] + r\_new[i + 1];

    }

    //виводимо розмір і сам масив

    cout << M << endl;

    for (int i = 0; i < M; i++) {

        cout << M\_sum[i] << " ";

    }

    return 0;

}

**5)Результати виконання завдань та фактично затрачений час**

Task 3 - Lab# programming: VNS Lab 4(варіант 4)

Зображення, що містить текст, знімок екрана, Шрифт

Автоматично згенерований опис

Фактично затрачений час: 32хв

Task 4 - Lab# programming: VNS Lab 5(варіант 4)

Зображення, що містить текст, Шрифт, знімок екрана, типографія

Автоматично згенерований опис

Фактичний час затрачений на виконання: 41хв

Task 5 - Lab# programming: Algotester Lab 2(варіант 1)

Зображення, що містить текст, знімок екрана, Шрифт

Автоматично згенерований описФактичний час затрачений на виконання: 23хв

Task 6 - Lab# programming: Algotester Lab 3(варіант 3)

Зображення, що містить текст, знімок екрана, Шрифт

Автоматично згенерований опис

Фактичний час затрачений на виконання: 18хв

Task 7 - Practice# programming: Class Practice Task

Зображення, що містить текст, знімок екрана, Шрифт

Автоматично згенерований опис

Фактичний час виконання: 49хв

Task 8  - Practice# programming:  Self Practice Task

Зображення, що містить текст, знімок екрана, Шрифт

Автоматично згенерований опис

Фактичний час виконання: 38хв

**6)Робота з комадою**

Trello:

Зображення, що містить текст, знімок екрана, програмне забезпечення, Мультимедійне програмне забезпечення

Автоматично згенерований опис

**Висновок:** Опановуючи роботу з масивами різних типів, вказівниками, посиланнями, динамічними структурами даних та алгоритмами їх обробки, ми набуваємо необхідних знань і навичок для ефективного управління даними та їхньої оптимізації. Це дозволяє будувати більш гнучкі та продуктивні програми, здатні працювати з великими обсягами даних, зберігаючи при цьому структурованість і логічну цілісність інформації.

**Посилання на пул реквест:** [**https://github.com/artificial-intelligence-department/ai\_programming\_playground\_2024/pull/269**](https://github.com/artificial-intelligence-department/ai_programming_playground_2024/pull/269)