Міністерство освіти і науки України
Національний університет «Львівська політехніка»
Кафедра систем штучного інтелекту

Звіт

про виконання лабораторних та практичних робіт блоку № 4
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-12
Бісюк Роман Васильович

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:

Дослідження одновимірних і двовимірних масивів для зберігання і впорядкування даних, що забезпечує швидкий доступ і обробку великих обсягів інформації.

Дослідження вказівників та посилань для розуміння адресації пам’яті та оптимізації використання ресурсів, що дозволяє ефективніше працювати з динамічними структурами даних.

Дослідження динамічних масивів для створення програм із змінною кількістю елементів, що підвищує гнучкість і адаптивність коду.

Дослідження структур даних та вкладених структур для організації складних об'єктів, що забезпечує кращу структуру і читабельність програмного коду.

Дослідження алгоритмів обробки масивів і структур для реалізації ефективної обробки даних, що сприяє написанню оптимізованих і масштабованих програм.

Теоретичні відомості:

У даній роботі розглядаються основні принципи роботи з масивами та структурами даних, зокрема одновимірні й двовимірні масиви для організації і зберігання великих обсягів даних. Особливу увагу приділено вказівникам і посиланням як засобам управління пам'яттю та ефективного доступу до даних. Розглянуто динамічні масиви, які забезпечують гнучке управління розміром даних під час виконання програми. Досліджено основи структур даних і вкладених структур для створення складних, логічно організованих об’єктів. Описано алгоритми обробки масивів і структур, що дозволяють ефективно виконувати операції пошуку, сортування і модифікації даних, покращуючи оптимізацію коду.

Джерела:

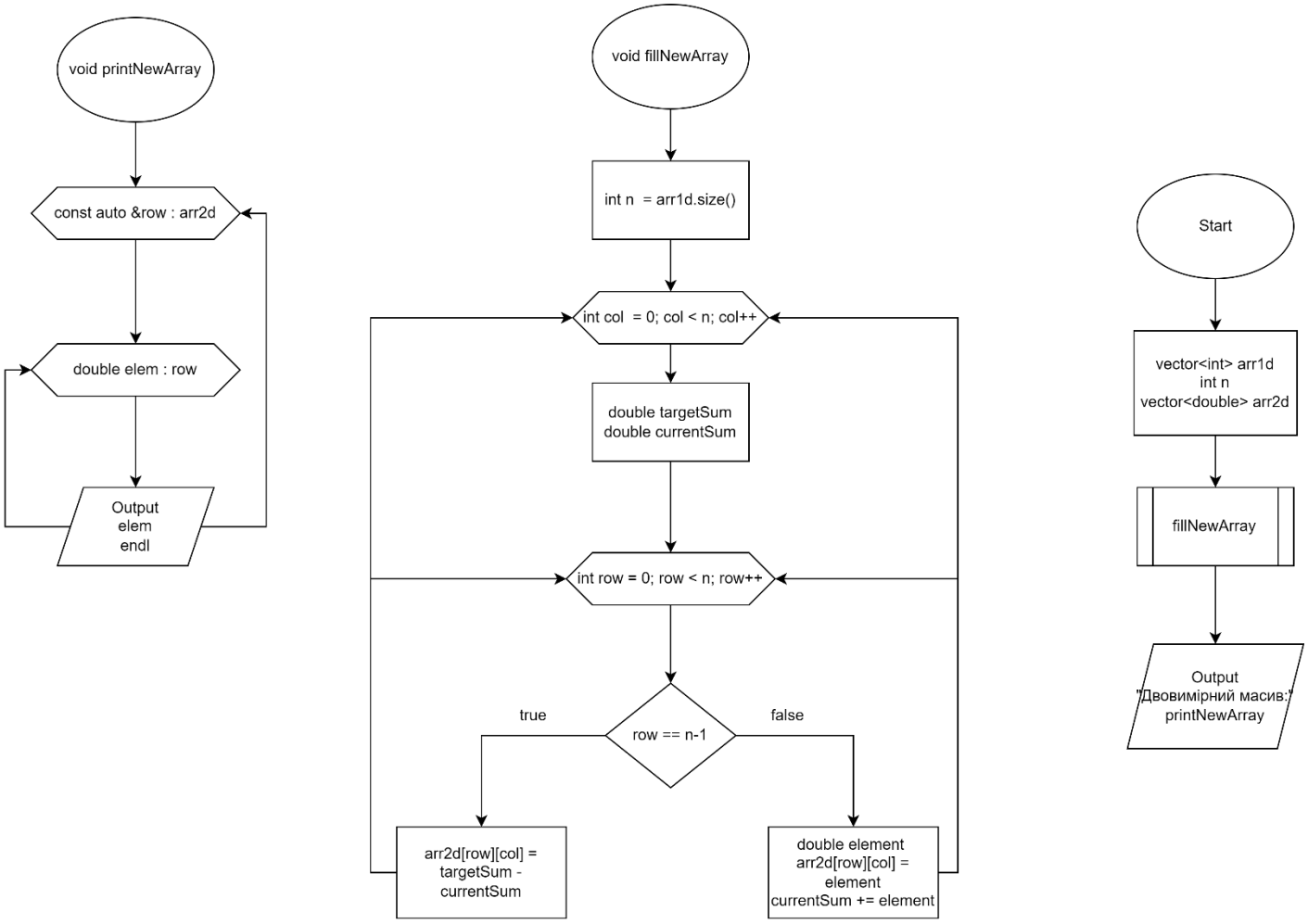
-aCode

- Harvard CS50 lectures+tasks

-w3school

Epic 4 Task 2 - Requirements management (understand tasks) and design activities (draw flow diagrams and estimate tasks 3-8)

Time expected – 20 minutes
time spent – 20 minutes



Epic 4 Task 3 - Lab# programming: VNS Lab 4

Time expected – 1 hour
time spent – 1.5 hour

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

using namespace std;


void printFromK(const vector<int>& arr, int k, bool forward)
{
    int n = arr.size();
    for (int i = 0; i < n; i++)
    {
        int index = forward ? (k + i) % n : (n + k - i) % n;
        cout << arr[index] << " ";
    }
    cout << endl;
}

void deleteMax(vector<int>& arr)
{
    int maxEl = *max_element(arr.begin(), arr.end());
    arr.erase(remove(arr.begin(), arr.end(), maxEl), arr.end());
}

int main()
{
    vector<int> elements = {3, 5, 1, 5, 3, 4, 5, 9};
    int k = 2;

    cout << "Елементи від k-го вліво:\n";
    printFromK(elements, k, false);

    deleteMax(elements);

    cout << "Елементи від k-го вправо:\n";
    printFromK(elements, k, true);

    return 0;
}
```

```
Елементи від k-го вліво:
1 5 3 9 5 4 3 5
Елементи від k-го вправо:
1 5 3 4 5 3 5
```

Epic 4 Task 4 - Lab# programming: VNS Lab 5

Time expected - 1 hour
time spent – 50 minutes

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

void fillNewArray(const vector<int> &arr1d, vector<vector<double>> &arr2d)
{
    int n = arr1d.size();

    for (int col = 0; col < n; col++)
    {
        double targetSum = arr1d[col];
        double currentSum = 0;

        for (int row = 0; row < n; row++)
            if (row == n - 1)
                arr2d[row][col] = targetSum - currentSum;
            else
            {
                double element = (targetSum / n) + (0.1 * (row + 1));
                arr2d[row][col] = element;
                currentSum += element;
            }
    }
}

void printNewArray(const vector<vector<double>> &arr2d)
{
    for (const auto &row : arr2d)
    {
        for (double elem : row)
            cout << elem << "\t";
        cout << endl;
    }
}

int main()
{
    vector<int> arr1d = {14, 23, 232, 4};
    int n = arr1d.size();

    vector<vector<double>> arr2d(n, vector<double>(n, 0.0));

    fillNewArray(arr1d, arr2d);

    cout << "Двовимірний масив: \n";
    printNewArray(arr2d);

    return 0;
}
```

```
Двовимірний масив:
3.6     5.85    58.1    1.1
3.7     5.95    58.2    1.2
3.8     6.05    58.3    1.3
2.9     5.15    57.4    0.4
```

Epic 4 Task 5 - Lab# programming: Algotester Lab 2

Time expected – 1 hour

Time spent – 40 minutes

Lab 2v3

Limits: 1 sec., 256 MiB

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 a_1, 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 a_i та a_{i+1} - виведіть **Stopped**

Врахуйте, що перевіряти треба також до взльоту.

Input

У першому рядку ціле число N - розмір масиву

У другому рядку N цілих чисел - елементи масиву

Output

У першому рядку фінальна позиція першого та другого дрона.

У другому рядку одне зі слів:

Created	Compiler	Result	Time (sec.)	Memory (MiB)	Actions
8 hours ago	C++ 23	Accepted	0.003	1.441	View

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

using namespace std;

int main() {
    size_t size;
    cin >> size;

    vector<int> arr(size);
    for (size_t i = 0; i < size; i++) {
        std::cin >> arr[i];
    }

    int leftP = 0;
    int rightP = size - 1;
    string answer = "Miss";
    bool canMove = true;

    while (canMove) {
        if (leftP == rightP) {
            answer = "Collision";
            break;
        }
        if (leftP == rightP - 1) {
            answer = "Stopped";
            break;
        }

        leftP += arr[leftP];
        rightP -= arr[rightP];

        if (leftP > rightP) {
            answer = "Miss";
            break;
        }
    }

    cout << leftP + 1 << ' ' << rightP + 1 << '\n' << answer;
    return 0;
}
```

Epic 4 Task 6 - Lab# programming: Algotester Lab 3

Time expected – 40 minutes
time spent – 1 hour

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

using namespace std;

string compress_string(const string& s) {
    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.push_back(s[i - 1]);
            if (count > 1) {
                compressed += to_string(count);
            }
            count = 1;
        }
    }

    return compressed;
}

int main() {
    string s;
    cin >> s;

    string compressed = compress_string(s);
    cout << compressed << endl;

    return 0;
}
```

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

Created	Compiler	Result	Time (sec.)	Memory (MiB)	Actions
8 hours ago	C++ 23	Accepted	0.003	1.418	View

Epic 4 Task 7 - Practice# programming: Class Practice Task

Time expected – 1 hour
time spent – 1 hour

```cpp
#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) {
    int original = num;
    int reversed = 0;

    while (num != 0) {
        reversed = reversed * 10 + num % 10;
        num /= 10;
    }
    return original == reversed;
}

int main() {
    string word;
    cout << "Введіть слово для перевірки на паліндром: ";
    cin >> word;
    if (isPalindrome(word, 0, word.length() - 1)) {
        cout << word << " є паліндромом" << endl;
    } else {
        cout << word << " не є паліндромом" << endl;
    }

    int number;
    cout << "Введіть число для перевірки на паліндром: ";
    cin >> number;
    if (isPalindrome(number)) {
        cout << number << " є паліндромом" << endl;
    } else {
        cout << number << " не є паліндромом" << endl;
    }

    return 0;
}
```

```
Введіть слово для перевірки на паліндром: radar
radar є паліндромом
Введіть число для перевірки на паліндром: 12345
12345 не є паліндромом
```

Epic 4 Task 8 - Practice# programming: Self Practice Task

Time expected - 30 minutes
time spent - 40 minutes

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

using namespace std;

int main() {
    int n;
    cin >> n;
    vector<pair<int, int>> offices(n);

    for (int i = 0; i < n; i++) {
        int length;
        cin >> length;
        offices[i] = {length, i + 1};
    }

    sort(offices.begin(), offices.end());

    for (auto office : offices) {
        cout << office.second << " ";
    }
    cout << endl;

    return 0;
}
```

Офісна Вулиця. Частина 1

Limits: 2 sec., 256 MiB

Зустрілися якось працівники великих компаній і почали... Обговорювати план вулиці.

Виявляється, всі приміщення, які орендуватимуть ці компанії, збудують вздовж однієї вулиці.

i-та компанія орендуватиме офіс довжиною l_i метрів. Офіси будуватимуть один за одним, починаючи з точки 0. Всі працівники приїжджатимуть на стоянку, яку побудують в точці 0, та будуть йти до офісів своїх компаній.

Тобто, якщо офіси будуть збудовані в порядку $p_1, p_2, \ldots, p_n$, то перший офіс почнеться в точці 0 і закінчиться в точці l_{p_1}, другий почнеться в l_{p_1} і закінчиться в $l_{p_1} + l_{p_2}$ і т.д. Двері кожного офісу завжди є в кінці будинку, який є ближчим до стоянки.

Ваше завдання — допомогти розмістити офіси компаній на цій вулиці в такому порядку, щоб сумарна відстань від точки 0 до усіх офісів була мінімальною.

Input

У першому рядку задане ціле число n — кількість компаній.

У наступному рядку задано n цілих чисел l_i через пробіл — довжини офісів усіх компаній.

Output

У єдиному рядку виведіть n чисел від 1 до n — порядок компаній, в якому варто будувати офіси.

Якщо існує декілька оптимальних порядків — виведіть будь-який із них.

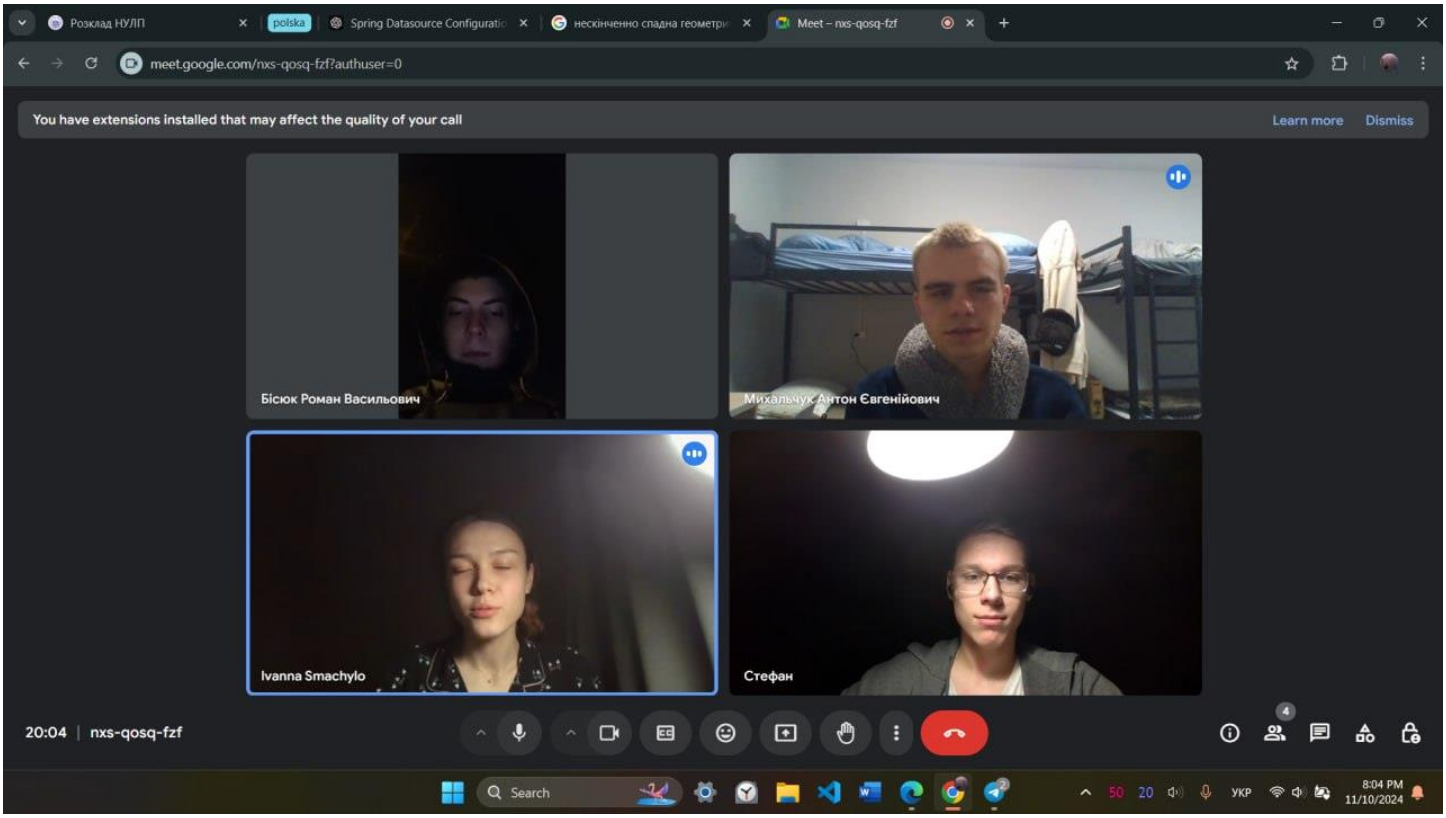
Created	Compiler	Result	Time (sec.)	Memory (MiB)	Actions
a few seconds ago	C++ 23	Accepted	0.053	2.273	View

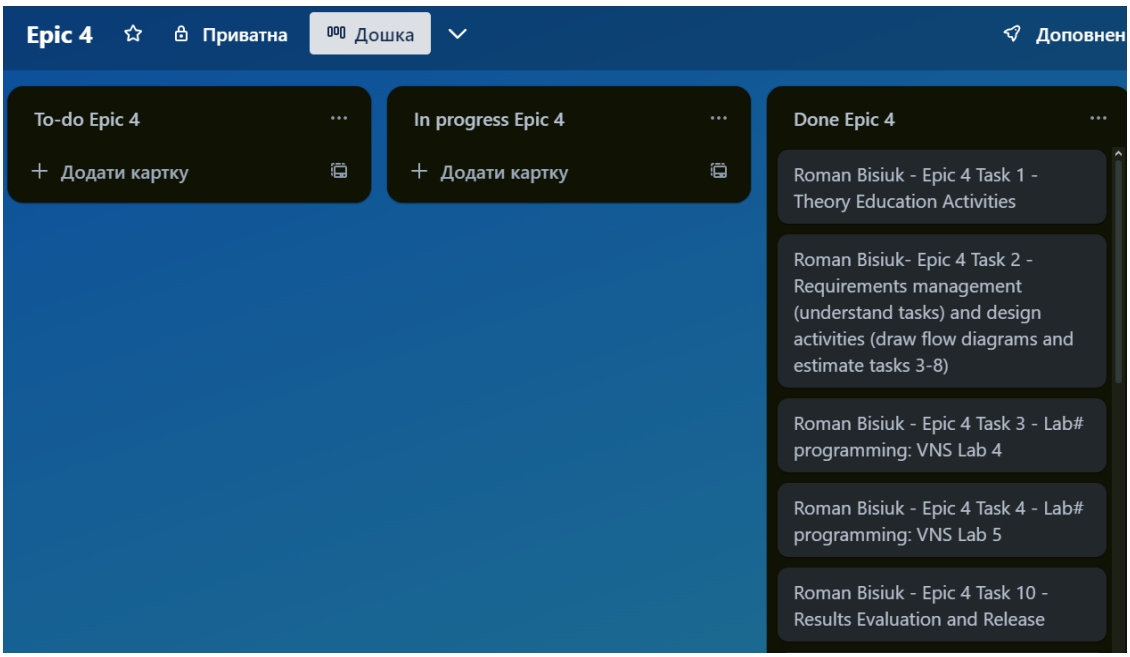
Epic 4 Task 9 - Result Documentation Report and Outcomes Placement Activities (Docs and Programs on GitHub)

Time expected - 30 minutes
time spent – 25 minutes

Pull-Request

Meets: розібралися з дошкою в Trello, обговорили проблемні питання





Висновок: В процесі виконання лабораторної роботи я навчився використовувати одновимірні та двовимірні масиви для зберігання і впорядкування даних, що покращує доступ до великого обсягу інформації. Також я ознайомився з поняттями вказівників і посилань, що дозволяє ефективно управляти пам’яттю і використовувати динамічні масиви. Окрім того, я досліджував структури даних та алгоритми обробки масивів, що сприяє написанню оптимізованих і масштабованих програм.