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

Звіт

про виконання лабораторних та практичних робіт блоку № 4

На тему: «Програмування: алгоритм, програма, код. Системи числення. Двійкова система числення. Розробка та середовище розробки програми.»

з дисципліни: «Основи програмування»

до:

Практичних Робіт до блоку № 4

Виконав:
Студент(ка) групи ШІ-13
Яцишин Роман Олегович

Львів 2024

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

Мета: Навчитись використовувати масиви, вказівники та посилання, організовувати структури даних. Засвоїти на практиці алгоритми обробки та роботи з масивами та структурами

Теоретичні відомості з переліком важливих тем:

1. Класи пам'яті у C++
    - Статична пам’ять.
    - Динамічна пам’ять.
    - Поняття стеку.
    - Виділення та вивільнення пам’яті.
2. 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.
3. 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.
4. 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.
5. 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.
6. 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.

7. 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)

8. 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
10. 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.

Використані джерела:

- https://www.w3schools.com/cpp/cpp_vectors.asp
- https://cplusplus.com/reference/vector/vector/
- https://cplusplus.com/reference/array/array/
- https://stackoverflow.com/questions/5590381/how-to-convert-int-to-string-in-c
- https://www.geeksforgeeks.org/cpp-recursion/
- https://www.geeksforgeeks.org/cpp-multidimensional-array/
- https://en.cppreference.com/w/cpp/string/byte/memcpy

Виконання роботи:

1. Опрацювання завдання та вимог до програм та середовища:

Програмний код №1

— Метою завдання є Сформувати одновимірний масив цілих чисел, використовуючи генератор випадкових чисел. Роздрукувати отриманий масив. Знищити елементи, індекси яких кратні 3. Додати після кожного від'ємного елемента масиву елемент зі значенням | M[I-1]+1|. Роздрукувати отриманий масив.
— Важливо було ураховувати синтаксис масиву, крок кратності.

Програмний код №2

— Метою завдання є визначити чи є матриця ортонормованою.
— Важливо, що скалярний добуток кожної пари різних рядків дорівнює 0, а скалярний добуток рядка самого на себе дорівнює 1.

Програмний код №3 Варіант 4

— Мета завдання: 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?

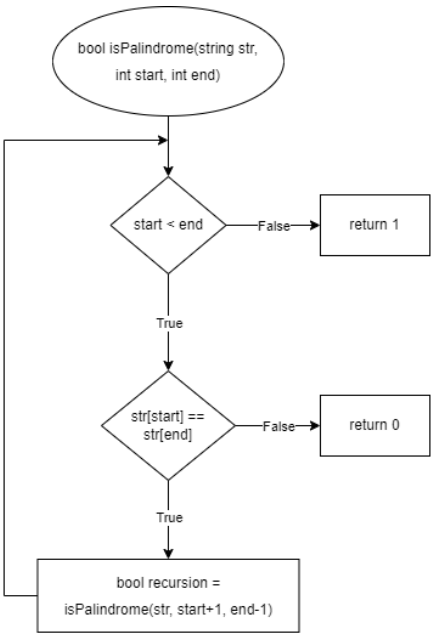
Програмний код №4 Варіант 4

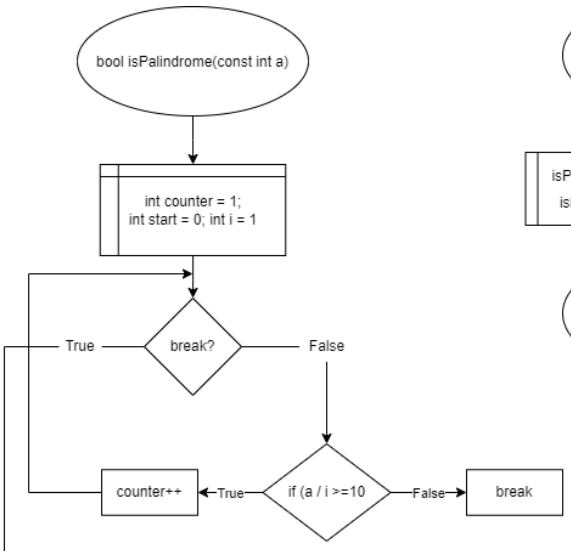
— Мета завдання: вам дана стрічка s. Ваше завдання зробити компресію стрічки, тобто якщо якась буква йде більше одного разу підряд у стрічці замінити її на букву + кількість входжень підряд.

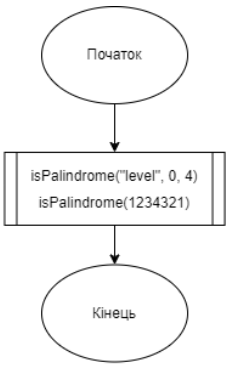
Програмний код №5

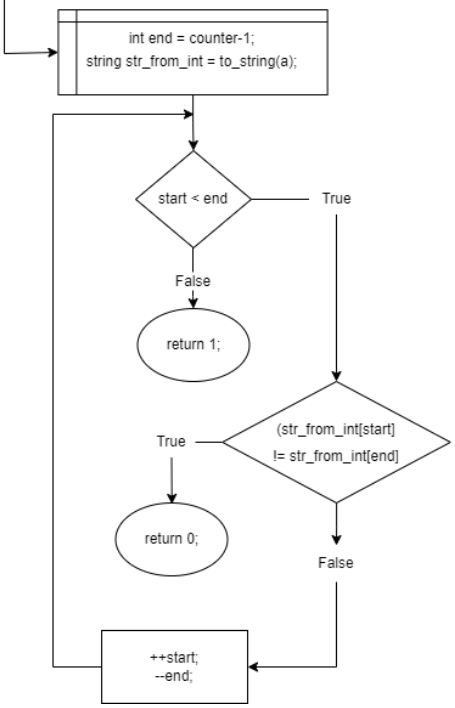
— Метою завдання було перевірити, чи слово/число є паліндромом.

2. Дизайн та планована оцінка часу виконання завдань:









3. Код програми:

VNS Lab 4

```cpp
#include <stdio.h>
#include <iostream>
#include <array>

using namespace std;

int main(){
    int myarray[10] = {-50, -60, -85, 66, 41, 2, -2, -6, 44, 4};
    // 1
    for (int i = 0; i < 10; i++)
    {
        // myarray[i] = rand() % 200 + -100;
        cout << myarray[i] << " ";
    }
    cout << "\n";

    // 3
    int newsize = 0;
    for (int i = 0; i < 10; i++)
    {
        if (i % 3 != 0)
        {
            myarray[newsize++] = myarray[i];
        }
    }

    int* myarray2 = new int[newsize];
```

```
    for (int i = 0; i < newsize; i++)
    {
        myarray2[i] = myarray[i];
    }
    for (int i = 0; i < newsize; i++) cout << myarray2[i] << " ";
    cout << endl;

    // 4

    for (int i = 0; i < newsize; )
    {
        if (myarray2[i] < 0)
        {
            int* newArr = new int[newsize + 1];
            memcpy(newArr, myarray2, (i+1) * sizeof(int));
            newArr[i+1] = myarray2[i]+1;
            memcpy(newArr + (i+1) + 1, myarray2 + (i+1), (newsize - (i+1)) * sizeof(int));
            delete[] myarray2;
            myarray2 = newArr;
            ++newsize;
            i += 2;
        }
        else{
            ++i;
        }

    }
    for (int i = 0; i < newsize; i++) cout << myarray2[i] << " ";
            cout << endl;

    delete[] myarray2;
}
```

VNS Lab 5

```cpp
#include <stdio.h>
#include <iostream>

using namespace std;

// n - розмірність матриці

void checker_for_three(int n, int a[3][3]){
    int s, self_sum = 0;
    bool self_s;
    // перший і другий
    for (int j = 0; j < n; j++)
    {
        s += a[0][j] * a[1][j];
    }
    // cout << s << '\n';

    // перший і третій
    for (int j = 0; j < n; j++)
    {
        s += a[0][j] * a[2][j];
    }
    cout << s << '\n';

    // другий і третій
    for (int j = 0; j < n; j++)
    {
        s += a[1][j] * a[2][j];
    }
    // cout << s << '\n';
```

```
    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < n; j++)
        {
            self_sum = a[i][j] * a[i][j];
        }
        if (self_sum == 1)
        {
            self_s = true;
        }
        else {self_s = false;}

    }
    // cout << self_s << '\n';


    if ((s != 0) || (self_s != true)) {cout << "Не ортогоналізована";}
    else if ((s == 0) && (self_s == true)) {cout << "Ортогоналізована";}

}

void checker_for_four(int n, int a[4][4]){
    int s, self_sum = 0;
    bool self_s;
    // перший і другий
    for (int j = 0; j < n; j++)
    {
        s += a[0][j] * a[1][j];
    }

    // перший і третій
    for (int j = 0; j < n; j++)
    {
        s += a[0][j] * a[2][j];
    }
```

```
    // перший і четвертий
    for (int j = 0; j < n; j++)
    {
        s += a[0][j] * a[3][j];
    }

    // другий і третій
    for (int j = 0; j < n; j++)
    {
        s += a[1][j] * a[2][j];
    }

    // другий і четвертий
    for (int j = 0; j < n; j++)
    {
        s += a[1][j] * a[3][j];
    }

    // третій і четвертий
    for (int j = 0; j < n; j++)
    {
        s += a[2][j] * a[3][j];
    }

    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < n; j++)
        {
            self_sum = a[i][j] * a[i][j];
        }
        if (self_sum == 1)
        {
            self_s = true;
        }
        else {self_s = false;}

    }
```

```
    if ((s != 0) || (self_s != true)) {cout << "Не ортогоналізована";}
    else if ((s == 0) && (self_s == true)) {cout << "Ортогоналізована";}

}

int main(){
    int b[4][4] = {
        {1, 0, 0, 0},
        {0, 1, 0, 0},
        {0, 0, 1, 0},
        {0, 0, 0, 1}};
    int c[3][3] = {
        {0, 1, 0},
        {1, 0, 0},
        {0, 0, -1}};

    checker_for_four(4, b);
    checker_for_three(3, c);
}
```

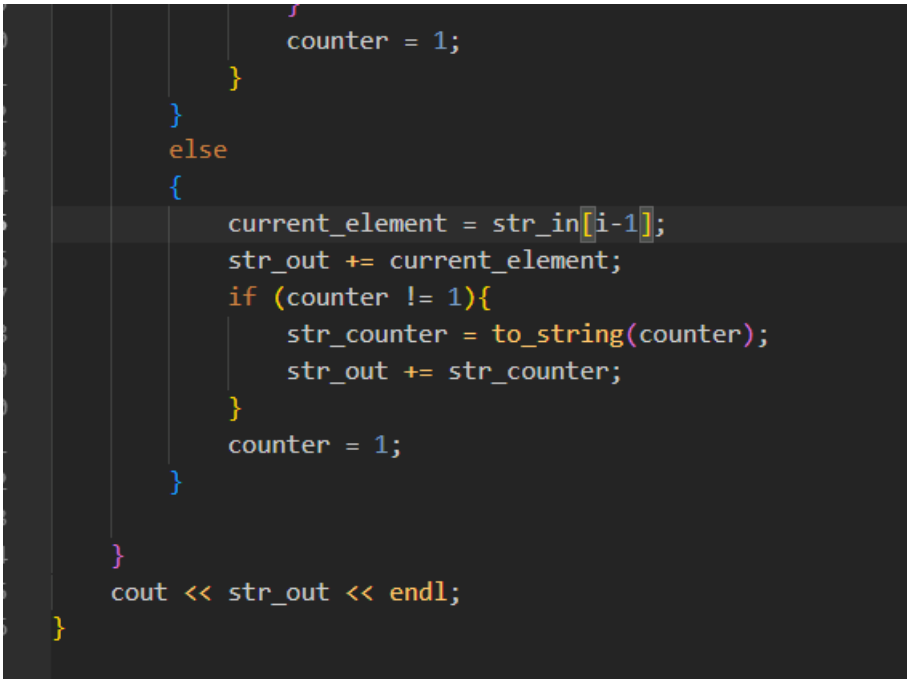
Algotester Lab 2

```cpp
#include <stdio.h>
#include <cmath>
#include <iostream>
#include <stdarg.h>
#include <vector>

using namespace std;

int func(int n, int i, int dif_old, vector<int> inputs){
    int differ = 0;

    while (i < n)
    {
        int current_element = inputs.at(i);

        inputs.erase(inputs.begin() + i);

        for (int a = 0; a < n-1; a++)
        {
            for (int b = 0; b < n-1; b++)
            {
                if (abs(inputs.at(a)-inputs.at(b)) > differ)
                {
                    differ = abs(inputs.at(a)-inputs.at(b));
                }
            }
        }

        inputs.insert(inputs.begin() + i, current_element);

        if (differ < dif_old) {dif_old = differ;}

        int recursive = func(n, ++i, dif_old, inputs);
        return recursive;
    }

    return dif_old;
}
```

```cpp
int main(){
    int element = 0;

    int n = 0;
    cin >> n;

    vector<int> inputs;

    for (int c = 0; c < n; c++) {
        cin >> element;
        inputs.push_back(element);
    }

    int i = 0;
    int dif_old = 1e6;
    int dif = func(n, i, dif_old, inputs);

    cout << dif;

    return 0;
}
```

Algotester Lab 3

```cpp
#include <stdio.h>
#include <iostream>
#include <string>

using namespace std;

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

    string str_out;
    int counter = 1;
    char current_element;
    string str_counter;

    // cout << str_in.length() << "\n";

    for (int i = 1; i <= str_in.length(); i++)
    {
        if (str_in.length() == 1)
        {
            str_out += str_in[0];
            break;
        }

        if (i != str_in.length())
        {
            current_element = str_in[i-1];
            if (str_in[i-1] == str_in[i])
            {
                ++counter;
            }
            else
            {
                str_out += current_element;
                if (counter != 1){
                    str_counter = to_string(counter);
                    str_out += str_counter;
                }
```



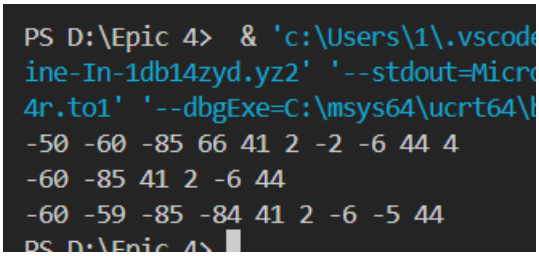
Class work Practice

```cpp
#include <stdio.h>
#include <iostream>
#include <string>
#include <cmath>

using namespace std;

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

    if (start < end) {
        if (str[start] == str[end])
        {
            bool recursion = isPalindrome(str, start+1, end-1);
            return recursion;
        }
        else
        {
            return 0;
        }
    }
    else{
        return 1;
    }
}

bool isPalindrome(const int a){
    int counter = 1;

    int start = 0;

    for (int i = 1; ; i*=10)
    {
        if (a / i >= 10)
        {
            counter++;
        }
        else{
            break;
        }
```

```
    int end = counter-1;
    string str_from_int = to_string(a);

    while (start < end) {
        if ((str_from_int[start] != str_from_int[end]))
        {
            return 0;
        }
        ++start;
        --end;
    }
    return 1;
}

int main(){
    cout << isPalindrome("level", 0, 4);
    cout << isPalindrome(1234321);
    cout << isPalindrome("bob", 0, 2);
    cout << isPalindrome(1231);
    cout << isPalindrome("manipulation", 0, 11);
    cout << isPalindrome(55);
}
```

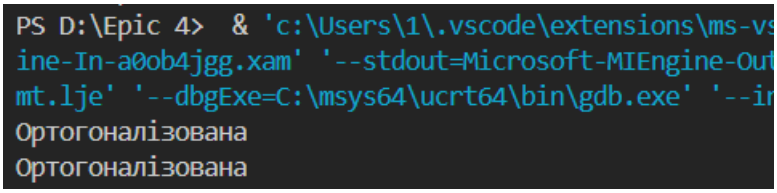
5. Результати виконання завдань, тестування та фактично затрачений час:

VNS Lab 4



Фактично витрачений час – 1 год

VNS Lab 5



Фактично витрачений час – 1.5 год

Algotester Lab 2

```
PS D:\Epic 4>  & 'c:\User
ine-In-4i13isy5.30u' '--s
nw.zr0' '--dbgExe=C:\msys
5
1 2 2 4 4
2
```

Фактично витрачений час – 2 год

Algotester Lab 3

```
PS D:\Epic 4>  & 'c:\Users\1\.vscoc
ine-In-ztrbwx12.frk' '--stdout=Micr
yb.ea3' '--dbgExe=C:\msys64\ucrt64\
aaaabbbcc
a4b3c2
PS D:\Epic 4>
```

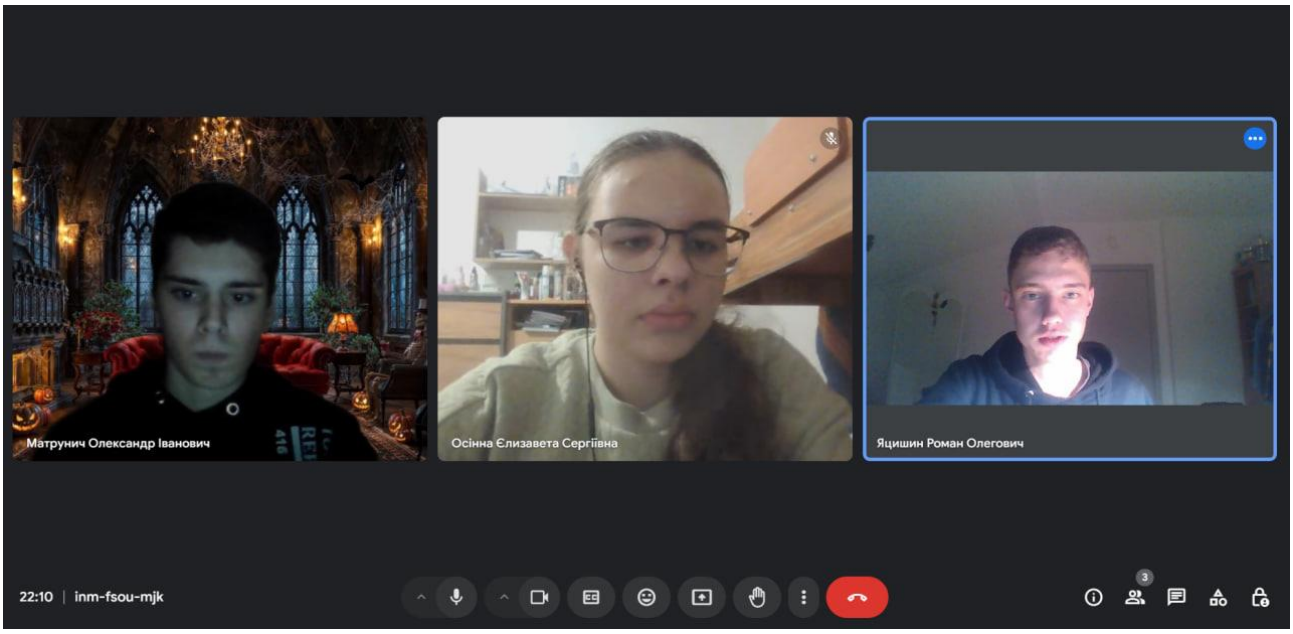
Фактично витрачений час – 1.5 год

Class work practice

```
PS D:\Epic 4>  & 'c:\Use
ine-In-vvbupxjk.aqb' '--
q0.kaw' '--dbgExe=C:\msy
111001
PS D:\Epic 4>
```

Фактично витрачений час – 1 год

6. Кооперація з командою:



Висновок:

Під час виконання цієї роботи було засвоєно основні принципи роботи з одновимірними та двовимірними масивами, вказівниками, посиланнями, динамічними масивами, а також зі структурами даних та вкладеними структурами. Використання масиви та вказівники для ефективного доступу до пам’яті, як статичне і динамічне виділення пам'яті допомагають управляти ресурсами програми.

Освоєно важливість структур для організації та обробки складних даних. Практика з алгоритмами сортування та пошуку для масивів і структур показала, як вони можуть пришвидшити обробку даних і спростити написання коду.