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

Звіт

про виконання лабораторних та практичних робіт блоку № 1-6
з дисципліни: «Мови та парадигми програмування»

Виконавла:
Студентка групи ШІ-13
Лемішко Марта Русланівна

Львів 2023

Тема роботи:

Епік 1: Розробка, програмування та код. Середовища для розробки.

Епік 2: Лінійні алгоритми. Розгалужені алгоритми. Умовні та логічні оператори. Системи числення. Змінні. Константи. Типи даних. Розмір Типів Даних (Двійкова система). Ввід вивід. Базові операції та вбудовані функції. Коментарі.

Епік 3: Цикли. Вкладені Цикли. Завершення виконання циклів. Функції. Простір імен. Перевантаження функцій. Функції з змінною кількістю параметрів (еліпсис). Рекурсія. Вбудовані функції.

Епік 4: Одновимірні масиви. Двовимірні Масиви. Вказівники та Посилання. Динамічні масиви. Структури даних. Вкладені структури. Алгоритми обробки та робота з масивами та структурами.

Епік 5: Файли. Бінарні Файли. Символи і Рядкові Змінні та Текстові Файли. Стандартна бібліотека та деталі/методи роботи з файлами. Створення й використання бібліотек.

Епік 6: Динамічні структури (Черга, Стек, Списки, Дерево). Алгоритми обробки динамічних структур.

Мета роботи:

Епік 1: Мета роботи - створити просту програму та налаштувати зручне середовище для її розробки. Це означає написання коду та налаштування інструментів для зручності роботи розробника.

Теоретичні відомості:

1) Теоретичні відомості з переліком важливих тем:
   - *Епік 1:*
     - Тема №1; Trello
     - Тема №2:Visual Studio Code
     - Тема №3: Git and GitHub
2) Індивідуальний план опрацювання теорії :
   a) *Епік 1:*
   - *Тема №1*: Trello.
     - Джерела Інформації
       - Стаття: https://trello.com/guide/create
       - Відео: https://www.youtube.com/watch?v=7voOifIpGJg
     - Що опрацьовано: Опрацьовано основні можливості платформи Trello для управління проектами та завданнями, включаючи створення дошок, списків та карток для організації робочих процесів. Також вивчено інтегроване середовище розробки Visual Studio Code, включаючи роботу з файлами, розширення та базові операції редагування коду. У темі Git and GitHub розглянуто базові команди для роботи з системою контролю версій, створення та об'єднання гілок, а також використання GitHub для збереження та обміну змінами в коді.
     - Статус: Ознайомлена
     - Початок опрацювання теми: 9.12.2023
     - Звершення опрацювання теми: 9.12.2023

Виконання роботи:

1. Опрацювання завдання та вимог до програм та середовища:

Завдання №1 Algotester Lab 1v1

- Варіант завдання 1
- Деталі завдання:

1. У персонажа є H хітпойнтів та M мани.
2. Персонаж використовує 3 закляття.
3. Кожне закляття може використовувати хітпойнти та ману одночасно.
4. Якщо якесь закляття забирає і хітпойнти, і ману - персонаж програє.
5. Для виграшу потрібно використовувати при одному заклинанні АБО хітпойнти, АБО ману.
6. У кінці гри, якщо персонаж має додатню кількість хітпойнтів та мани (H, M > 0), він виграє.
7. Завдання: вивести YES у випадку виграшу персонажа, вивести NO у іншому випадку.

Завдання №2 Algotester Lab 1v2

Задача вирішується шляхом визначення, чи стіл залишиться цілим та паралельним підлозі після відпилювання частин ніжок. Використовуючи вхідні дані щодо довжин кожної з чотирьох ніжок і кількість відпилення (d) для кожної, можна визначити, чи виконуються умови стабільності.

1. Зчитати чотири довжини ніжк (L1, L2, L3, L4) та кількість відпилення (d).
2. Знайти мінімальну та максимальну довжину ніжок.
3. Перевірити умову для перевертання стола: **max >= 2 * min**.
4. Якщо умова перевертання виконується, вивести "ERROR" (стіл перевернеться).
5. Якщо умова не виконується, відпиляти вказану кількість (d) від кожної ніжки.
6. Перевірити, чи довжина, яку відпиляють, менша за довжину ніжки. Якщо так, вивести "ERROR".
7. Якщо усі перевірки пройдені, вивести "YES" (стіл залишиться цілим та паралельним підлозі).

Завдання №3 Algotester Lab 2v1

Для мінімізації втоми потрібно видалити один елемент із дороги так, щоб різниця між новим максимальним та мінімальним елементами була мінімальною. Це можна зробити, порівняючи ефект видалення кожного елемента.

1. Зчитати дорогу як масив з N чисел.
2. Знайти максимальний та мінімальний елемент в дорозі.
3. Для кожного елемента дороги зробити наступне: a. Видалити поточний елемент та знайти новий максимальний та мінімальний елемент. b. Розрахувати різницю між новим максимальним та мінімальним елементами.
4. Знайти мінімальну різницю серед усіх обчислених значень.
5. Вивести мінімальну різницю.

Завдання №4 Algotester Lab 2v2

1. Зчитати масив **r** розміром **N**.
2. Зчитати три цілі числа, які слід видалити з масиву.

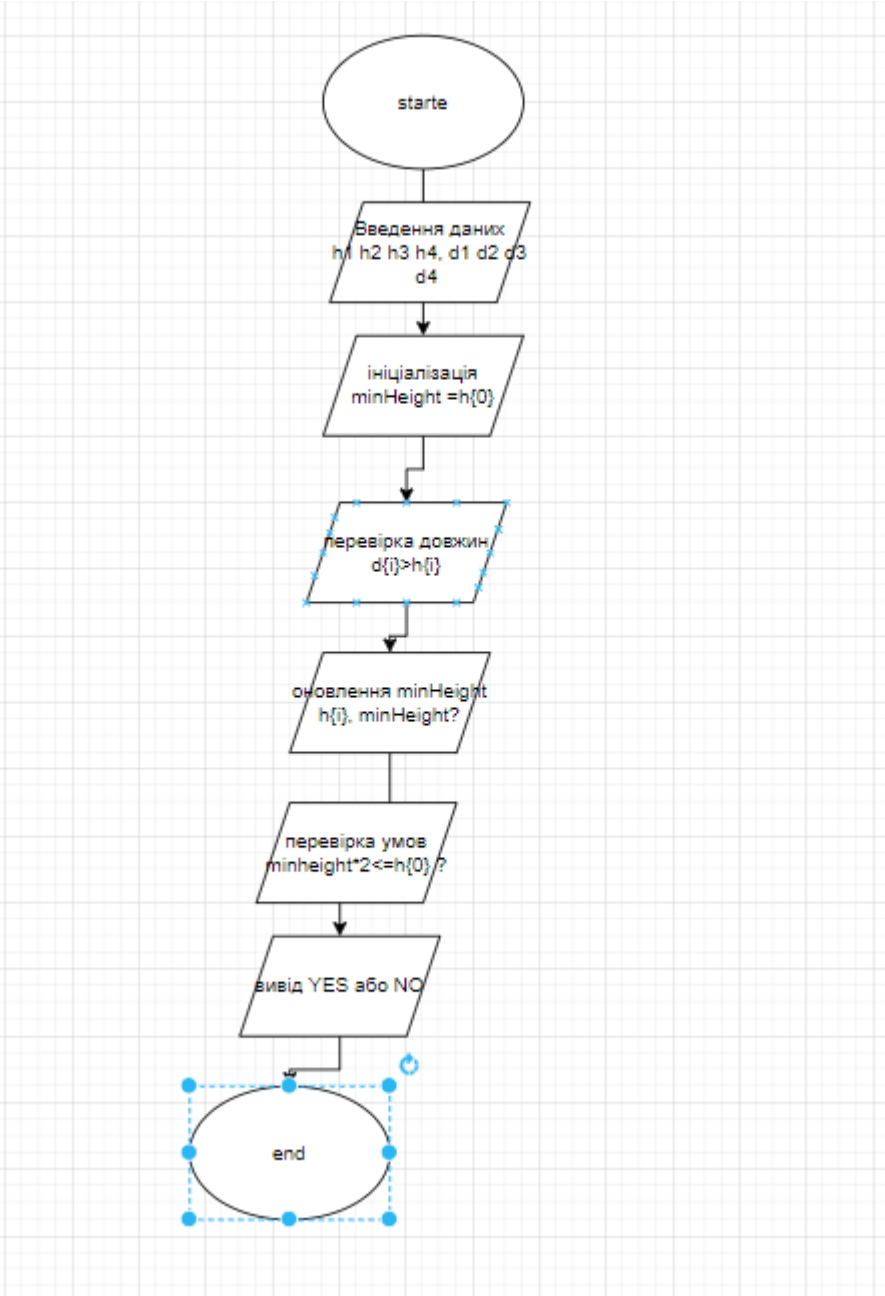
3. Видалити ці три числа з масиву **r**.
4. Створити новий масив сум, **sums**, розміром **N - 1**.
5. Для кожного індексу **i** від 0 до **N - 2** присвоїти **sums[i]** значення **r[i] + r[i+1]**.
6. Вивести масив сум **sums** на екран.

Завдання №5 Algotester Lab 3v3

1. Оголошення стрічкових змінних та зчитування введеної стрічки.
2. Перевірка допустимого розміру стрічки.
3. Ініціалізація лічильника та цикл проходження через символи стрічки.
4. Підрахунок кількості входжень кожного символу підряд.
5. Додавання стиснутого символу та кількості входжень до нової стрічки.
6. Вивід стиснутої стрічки.

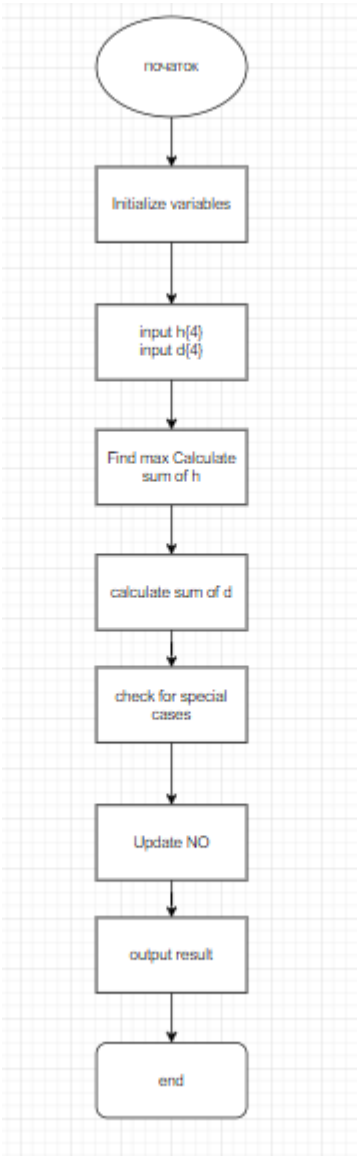
2. Дизайн та планована оцінка часу виконання завдань:

Програма №1 Algotester Lab 1v1



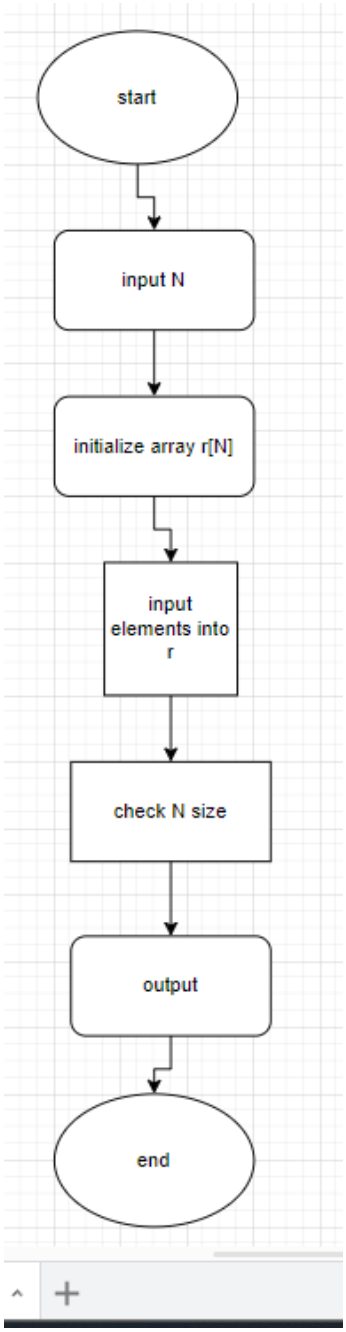
Планований час на реалізацію:1 день

Програма №2 Algotester Lab 1v2

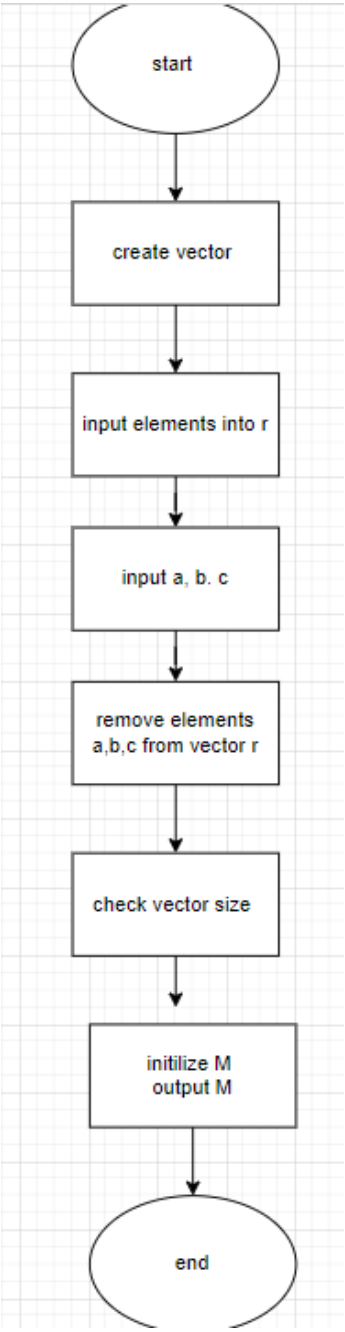


Планований час на реалізацію:1 день

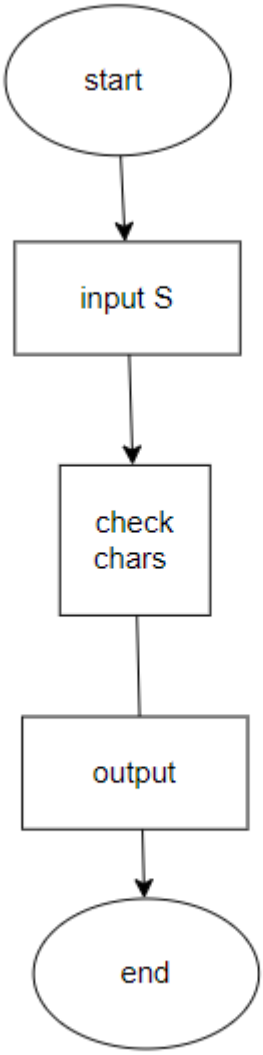
Програма №3 Algotester Lab 2v1
Планований час на реалізацію:2 день



Програма №4 Algotester Lab 2v2
Планований час на реалізацію:1 день



Програма №5 Algotester Lab 3v3
Планований час на реалізацію:2 день



3. Конфігурація середовища до виконання завдань:

- Додаткова конфігурація середовища не потрібна

4. Код програм з посиланням на зовнішні ресурси:

Завдання №1 Algotester Lab 1v1
https://github.com/artificial-intelligence-department/ai_programming_playground/pull/1390

Завдання №2 Algotester Lab 1v2

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

using namespace std;

int main() {
    long long h[4], max = 0, flag = 0, sum = 0, sum1 = 0;
    long long d[4];
    string rez = "YES";

    for (int i = 0; i < 4; i++) {
        cin >> h[i];
        if (h[i] > max) {
            max = h[i];
        }
        sum += h[i];
    }

    for (int i = 0; i < 4; i++) {
        cin >> d[i];
        sum1 += d[i];
    }

    if (sum + sum1 == 0 || sum1 == 0) {
        cout << "YES";
        return 0;
    }
```

https://github.com/artificial-intelligence-department/ai_programming_playground/pull/1390

Завдання №3 Algotester Lab 2v1

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

int main (){
    int m;
    long long N;
    cin >> N;
    long long r [N];
    for(int i = 0; i < N; i++)
     {
        cin >> r[i];
     }
    for (int i= 0; i< N; i++)
    {
        for (int  j=0; j< N-1-i; j++)
        {
            if (r[j]>r[j+1])
            {
                int temp=r[j];
                r[j]=r[j+1];
                r[j+1]= temp;
            }
        }
    }
    if (N==1)
    {
        cout << 0;
```

Завдання №4 Algotester Lab 2v2

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

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

    std::vector<int> r(N);
    for (int i = 0; i < N; ++i) {
        std::cin >> r[i];
    }

    int a, b, c;
    std::cin >> a >> b >> c;

    r.erase(std::remove_if(r.begin(), r.end(), [a, b, c](int x) {
        return x == a || x == b || x == c;
    }), r.end());

    if (r.size() < 2) {
        std::cout << "0" << std::endl;
        return 0;
    }

    std::vector<int> sum_array;
    for (int i = 0; i < r.size() - 1; ++i) {

    std::vector<int> sum_array;
    for (int i = 0; i < r.size() - 1; ++i) {
        sum_array.push_back(r[i] + r[i + 1]);
    }

    int M = sum_array.size();
    std::cout << M << std::endl;
    for (int i = 0; i < M; ++i) {
        std::cout << sum_array[i] << " ";
    }

    return 0;
}
```

Завдання №5 Agotester Lab 3v3

```
        cout << s_compressed << endl;
    }

    return 0;
}
```

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

using namespace std;

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

    if (s.size() > 0 && s.size() <= 100000) {
        int number = 1;

        for (int i = 0; i < s.size(); i++) {
            if (i + 1 < s.size() && s[i] == s[i + 1]) {
                number++;
            } else {
                s_compressed += s[i];
                if (number > 1) {
                    s_compressed += to_string(number);
                }
                number = 1;
            }
        }

        cout << s_compressed << endl;
    }
```

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

Завдання №1 Algotester Lab 1v1

Створено	Компілятор	Результат	Час (сек.)	Пам'ять (МіБ)	Дії
8 днів тому	C++ 20	Зараховано	0.004	3.281	Перегляд

Час затрачений на виконання завдання 1 день

Завдання №2 Algotester Lab 1v2

Створено	Компілятор	Результат	Час (сек.)	Пам'ять (МіБ)	Дії
4 дні тому	C++ 20	Зараховано	0.003	2.117	Перегляд

Час затрачений на виконання завдання 1 день

Завдання №3 Algotester Lab 2v1

Створено	Компілятор	Результат	Час (сек.)	Пам'ять (МіБ)	Дії
4 дні тому	C++ 20	Зараховано	0.003	2.117	Перегляд

Час затрачений на виконання завдання 2 день

Завдання №4 Algotester Lab 2v2

Створено	Компілятор	Результат	Час (сек.)	Пам'ять (МіБ)	Дії
8 днів тому	C++ 20	Зараховано	0.004	3.207	Перегляд

Час затрачений на виконання завдання 1 день

Завдання №5 Algotester Lab 3v3

Створено	Компілятор	Результат	Час (сек.)	Пам'ять (МіБ)	Дії
8 днів тому	C++ 20	Зараховано	0.004	3.281	Перегляд

Час затрачений на виконання завдання 2 день

6. Кооперація з командою:

https://www.w3schools.com/

Висновок:

- **Цикли та умови:** Розуміння використання циклів та умовних конструкцій для керування виконанням програми. Це допомагає створювати більш гнучкі та ефективні програми.
- **Оператори масивів та векторів:** Вивчення використання масивів та векторів для обробки послідовностей даних, що є важливим у багатьох аспектах програмування.
- **Функції:** Розуміння використання функцій для розділення коду на логічні частини, що полегшує розробку та управління кодом.

- **Важливість правильної документації:** Звернення уваги на коментування коду, яке полегшує розуміння іншим програмістам або власному себе при подальшій роботі з кодом.
- **Пошук помилок:** Розуміння важливості відлагодження коду для знаходження та виправлення помилок.

Ці навички можуть бути корисними у різних сферах життя, де важливе програмування. . Вивчені навички дозволять ефективніше управляти та адаптувати програми під змінні умови, забезпечуючи надійну та оптимізовану роботу програмного забезпечення.

Навички контролю введення допоможуть уникнути некоректної обробки вхідних даних і підвищити надійність програм. Розуміння циклів та умовних конструкцій відкриває можливості для створення адаптивних програм, які взаємодіють з користувачем або оточенням. Використання масивів та векторів робить обробку даних більш ефективною та легше зрозумілою.

Функції сприяють створенню чистого та організованого коду, що полегшує його розвиток та супровід. Важливість документації і коментування коду стає ключовою для командної роботи та подальшого вдосконалення програм.

Навички пошуку та виправлення помилок роблять мене більш компетентною у підтримці і вдосконаленні існуючих програм, забезпечуючи їх стабільність та ефективність. Усе це виробляє базу для подальшого успішного розвитку у сфері програмування та використання отриманих навичок у практичних завданнях.