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 № 6**

На тему: «Динамічні структури (Черга, Стек, Списки, Дерево). Алгоритми обробки динамічних структур.»

***з дисципліни:*** «Мови та парадигми програмування»

до:

ВНС Лабораторної Роботи № 10

Алготестер Лабораторної Роботи № 5

Алготестер Лабораторної Роботи № 7-8

Практичних Робіт № 6

**Виконав:**

Студент групи ШІ-11

Стожук Роман Ігорович

# **Тема роботи:**

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

# **Мета роботи:**

# Навчитися працювати з динамічними структурами

# **Теоретичні відомості:**

1. Теоретичні відомості з переліком важливих тем:

* Тема №1: Динамічні структури
* Тема №2: Алгоритми обробки динамічних структур

1. Індивідуальний план опрацювання теорії:

* Тема №1: Динамічні структури
  + Джерела Інформації
    - https://acode.com.ua/urok-205-kontejnery-stl/
    - https://acode.com.ua/urok-206-iteratory-stl/
  + Що опрацьовано:
    - Роботу з чергами, стеками, списками, деревами
  + Статус: Ознайомлений
  + Початок опрацювання теми: 01.12.2023
  + Звершення опрацювання теми: 02.12.2023
* Тема №2: Алгоритми обробки динамічних структур
  + Джерела Інформації:
    - https://acode.com.ua/urok-207-algorytmy-stl/
  + Що опрацьовано:
    - Такі операції, як пошук, сортування, вставка, видалення і копіювання елементів контейнера.
  + Статус: Ознайомлений
  + Початок опрацювання теми: 02.12.2023
  + Звершення опрацювання теми: 03.12.2023

# **Виконання роботи:**

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

Завдання №3 VNS Lab 10

* Варіант завдання: 6
* 1.Написати функцію для створення списку. Функція може створювати порожній список, а потім додавати в нього елементи.

2. Написати функцію для друку списку. Функція повинна передбачати вивід повідомлення, якщо список порожній.

3. Написати функції для знищення й додавання елементів списку у відповідності зі своїм варіантом.

4. Виконати зміни в списку й друк списку після кожної зміни.

5. Написати функцію для запису списку у файл.

6. Написати функцію для знищення списку.

7. Записати список у файл, знищити його й виконати друк (при друці повинне бути видане повідомлення "Список порожній").

8. Написати функцію для відновлення списку з файлу.

9. Відновити список і роздрукувати його.

10.Знищити список.

* Сформувати двонаправлений список. Знищити з нього елемент із заданим номером, додати елемент у початок списку.

Завдання №4 Algotester Lab 5

* Варіант завдання: 2
* В пустелі існує незвичайна печера, яка є двохвимірною. Її висота це N, ширина - M. Всередині печери є пустота, пісок та каміння. Пустота позначається буквою О , пісок S і каміння X. Одного дня стався землетрус і весь пісок посипався вниз. Він падає на найнижчу клітинку з пустотою, але він не може пролетіти через каміння. Ваше завдання сказати як буде виглядати печера після землетрусу.

# Обмеження:

1≤N,M≤1000

|rowi|=M

Завдання №5 Algotester Lab 7-8

* Варіант завдання: 2

# Ваше завдання - власноруч реалізувати структуру даних "Динамічний масив". Ви отримаєте Q запитів, кожен запит буде починатися зі слова-ідентифікатора, після якого йдуть його аргументи. Вам будуть поступати запити такого типу:

* **Вставка**:  
  Ідентифікатор - insert  
  Ви отримуєте ціле число index елемента, на місце якого робити вставку.  
  Після цього в наступному рядку рядку написане число N - розмір масиву, який треба вставити.  
  У третьому рядку N цілих чисел - масив, який треба вставити на позицію index.
* **Видалення**:  
  Ідентифікатор - erase  
  Ви отримуєте 2 цілих числа - index, індекс елемента, з якого почати видалення та n - кількість елементів, яку треба видалити.
* **Визначення розміру**:  
  Ідентифікатор - size  
  Ви не отримуєте аргументів.  
  Ви виводите кількість елементів у динамічному масиві.
* **Визначення кількості зарезервованої пам’яті**:  
  Ідентифікатор - capacity  
  Ви не отримуєте аргументів.  
  Ви виводите кількість зарезервованої пам’яті у динамічному масиві.  
  Ваша реалізація динамічного масиву має мати фактор росту ([Growth factor](https://en.wikipedia.org/wiki/Dynamic_array#Growth_factor)) рівний 2.
* **Отримання значення** i-го елементу  
  Ідентифікатор - get  
  Ви отримуєте ціле число - index, індекс елемента.  
  Ви виводите значення елемента за індексом. Реалізувати використовуючи перевантаження оператора []
* **Модифікація значення** i-го елементу  
  Ідентифікатор - set  
  Ви отримуєте 2 цілих числа - індекс елемента, який треба змінити, та його нове значення. Реалізувати використовуючи перевантаження оператора []
* **Вивід динамічного масиву на екран**  
  Ідентифікатор - print  
  Ви не отримуєте аргументів.  
  Ви виводите усі елементи динамічного масиву через пробіл.  
  Реалізувати використовуючи перевантаження оператора <<

# Вхідні дані:

# Ціле число Q - кількість запитів. У наступних рядках Q запитів у зазначеному в умові форматі.

# Обмеження:

0≤Q≤100000  
0≤li≤100000  
∥l∥≤100000

# Завдання №6.1 Class Practice Task 1-3

## Реверс списку (Reverse list):

*Реалізувати метод реверсу списку:* Node\* reverse(Node \*head);

*Умови задачі:*

використовувати цілочисельні значення в списку;

реалізувати метод реверсу;

реалізувати допоміжний метод виведення вхідного і обернутого списків;

## Порівняння списків:

bool compare(Node \*h1, Node \*h2);

*Умови задачі:*

використовувати цілочисельні значення в списку;

реалізувати функцію, яка ітеративно проходиться по обох списках і порівнює дані в кожному вузлі;

якщо виявлено невідповідність даних або якщо довжина списків різна (один список закінчується раніше іншого), функція повертає *false*.

## Додавання великих чисел:

Node\* add(Node \*n1, Node \*n2);

*Умови задачі:*

використовувати цифри від 0 до 9 для значень у списку;

реалізувати функцію, яка обчислює суму двох чисел, які збережено в списку; молодший розряд числа записано в голові списка (напр. 379  ⟹  9→7→3);

функція повертає новий список, передані в функцію списки не модифікуються.

Завдання №6.2 Class Practice Task 4-5

## Віддзеркалення дерева:

TreeNode \*create\_mirror\_flip(TreeNode \*root);

*Умови задачі:*

використовувати цілі числа для значень у вузлах дерева;

реалізувати функцію, що проходить по всіх вузлах дерева і міняє місцями праву і ліву вітки дерева;

функція повертає нове дерево, передане в функцію дерево не модифікується;

## Записати кожному батьківському вузлу суму підвузлів:

void tree\_sum(TreeNode \*root);

*Умови задачі:*

використовувати цілочисельні значення у вузлах дерева;

реалізувати функцію, яка ітеративно проходить по бінарному дереві і записує у батьківський вузол суму значень підвузлів;

вузол-листок не змінює значення;

значення змінюються від листків до кореня дерева;

Завдання № 7.1 - Self Practice Task A

* Петрик любить грати в ігри, де світ складається з кубів. Сьогодні він вирішив написати свій варіант такої гри. Петрик хоче, щоб у цій грі можна було програмувати нескладні механізми на основі електричного сигналу. Вважається, що сигнал має певну додатну силу, яка позначається цілим числом. Також, сигнал може бути відсутнім, що позначається як сигнал силою 0. Петрик додав в гру три блоки, які по-різному обробляють електричний сигнал:
* Кабель отримує сигнал силою k і видає сигнал силою max(k−1,0).
* Повторювач підсилює сигнал до рівня 15 при ненульовому вхідному сигналі, і не видає сигнал при його відсутності. Формально, нехай сила вхідного сигналу k. Тоді повторювач видає сигнал силою 15, якщо k≥1, і сигнал силою 0, якщо k=0.
* Змінювач не видає сигнал при ненульовому вхідному сигналі та видає сигнал силою 15 при відсутності вхідного сигналу. Формально, нехай сила вхідного сигналу k. Тоді змінювач видає сигнал силою 0, якщо k≥1, і сигнал силою 15, якщо k=0.
* Петрик задає вам електричну лінію, в якій кабелі, повторювачі і змінювачі позначені літерами k, p, z відповідно. На перший елемент лінії подано сигнал силою 15. Допоможіть Петрику знайти вихідний сигнал останнього блоку в лінії.

Завдання № 7.2 - Self Practice Task 0921

* На свій день народження Зеник вирішив запросити Марічку до ресторану. Оскільки хотілося чогось вишуканого, його вибір впав на ресторан із цікавою назвою «Шаверма-Кебаб». Наївшись досхочу котятини, молоді люди вирішили покинути гостинне місце й узяли рахунок для оплати. Але чи то офіціант з’їв не тих грибів, чи то в цьому вишуканому закладі так прийнято, замість рахунку їм принесли аркуш паперу, на якому було написано два різні числа a й b. Головні герої нічого не зрозуміли, та й уже було темно за вікном, тому вони вирішили розрахуватися такою цілою сумою (у гривнях, без копійок), щоб вона була строго більшою за одне з цих чисел і строго меншою за інше. Скільки ж грошей вони могли заплатити?

# Вхідні дані:

У єдиному рядку задано два цілі числа a й b — числа, що були записані на аркуші паперу.

# Вихідні дані:

У єдиному рядку виведіть єдине ціле число c, яке є більшим за одне з чисел на аркуші й меншим за інше. Якщо таких чисел не існує, виведіть -1. Якщо таких чисел декілька, виведіть довільне.

# Обмеження:

1≤a,b≤2⋅10^9,

a≠b.

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

Програма №4 Algotester Lab 5

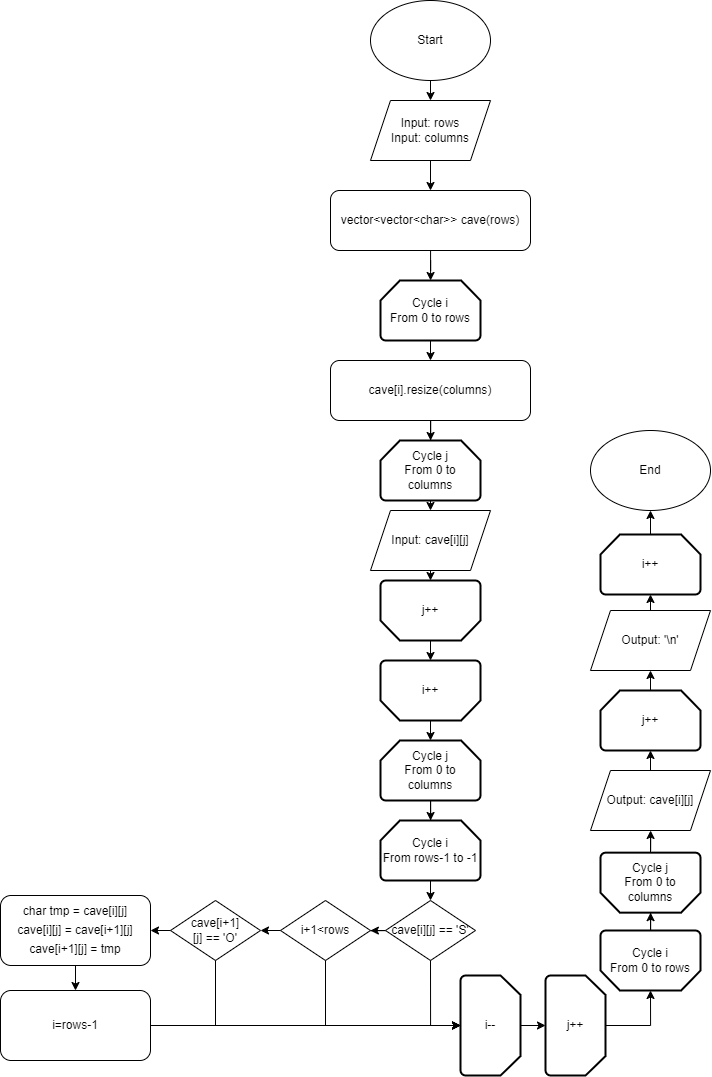
* 

Рисунок Блок схема до програми №4

* Планований час на реалізацію: 20хв

## **3. Конфігурація середовища до виконання завдань:**

## Змін у конфігурацію середовища не вносив.

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

Завдання №3 VNS Lab 10

#include <iostream>

#include <fstream>

using namespace std;

struct Node {

    int value=0;

    struct Node \*next=nullptr;

    struct Node \*prev=nullptr;

};

struct DLinkedList {

    unsigned long size=0;

    Node \*head=nullptr;

    Node \*tail=nullptr;

};

DLinkedList\* createList() {

    DLinkedList \*tmp = new DLinkedList;

    tmp->size=0;

    tmp->head=nullptr, tmp->tail=nullptr;

    return tmp;

}

void printList(DLinkedList \*\*list) {

    if(!(\*list) || !(\*list)->size) {cout << "List is empty";}

    else {

        Node \*tmp=(\*list)->head;

        while (tmp) {

            cout << tmp->value << ' ';

            tmp=tmp->next;

        }

    }

    cout << '\n';

}

void addToList(DLinkedList \*\*list, double number, int value) {

    Node \*toAdd=new Node;

    toAdd->value=value;

    toAdd->next=nullptr;

    toAdd->prev=nullptr;

    if (!((\*list)->size)) {

        (\*list)->head=toAdd;

        (\*list)->tail=toAdd;

    }

    else if (number == 1) {

        Node \*next=(\*list)->head;

        toAdd->next=next;

        next->prev=toAdd;

        (\*list)->head=toAdd;

    }

    else if (number-1 == (\*list)->size) {

        Node \*prev=(\*list)->tail;

        toAdd->prev=prev;

        prev->next=toAdd;

        (\*list)->tail=toAdd;

    }

    else if ((\*list)->size) {

        Node \*prev=(\*list)->head;

        Node \*next=(\*list)->tail;

        for (unsigned long i=1;i<number-1;i++) {

            prev=prev->next;

        }

        next=prev->next;

        toAdd->prev=prev;

        toAdd->next=next;

        prev->next=toAdd;

        next->prev=toAdd;

    }

    (\*list)->size++;

}

int deleteFromList(DLinkedList \*\*list, double number) {

    int delVal;

    Node \*toDelete=(\*list)->head;

    if ((\*list)->size==1) {

        delVal=toDelete->value;

        delete (\*list)->head;

        (\*list)->head=nullptr;

        (\*list)->tail=nullptr;

    }

    else if (number == 1) {

        delVal=toDelete->value;

        Node \*next=(\*list)->head->next;

        delete toDelete;

        (\*list)->head=next;

    }

    else if (number == (\*list)->size) {

        toDelete=(\*list)->tail;

        delVal=toDelete->value;

        Node \*prev=(\*list)->tail->prev;

        delete toDelete;

        (\*list)->tail=prev;

    }

    else if ((\*list)->size) {

        for (unsigned long i=0;i<number-1;i++) {

            toDelete=toDelete->next;

        }

        delVal=toDelete->value;

        Node \*prev=toDelete->prev;

        Node \*next=toDelete->next;

        delete toDelete;

        prev->next=next;

        next->prev=prev;

    }

    (\*list)->size--;

    return delVal;

}

void writeListToFile(DLinkedList \*\*list,char \*file) {

    ofstream out(file, ios::binary);

    Node \*current=(\*list)->head;

    for (unsigned long i=0; i<(\*list)->size; i++) {

        out << current->value << ' ';

        current=current->next;

    }

    out << '\n';

    out.close();

}

void deleteList(DLinkedList \*\*list) {

    Node \*tmp=(\*list)->head;

    if (tmp) { delete tmp; }

    (\*list)->size=0;

    delete (\*list);

    (\*list) = nullptr;

}

DLinkedList\* restoreListFromFile(char \*file) {

    ifstream in(file, ios::binary);

    DLinkedList \*restored=createList();

    Node \*add=nullptr, \*add\_next=nullptr;

    int tmp=0;

    in >> tmp;

    if(tmp) {

        add\_next = new Node;

        add\_next->next=nullptr;

        add\_next->prev=nullptr;

        add\_next->value=tmp;

        restored->head=add\_next;

        restored->head=add\_next;

        restored->size++;

    }

    while(1) {

        in >> tmp;

        if(in.eof()) {break;}

        add=add\_next;

        add\_next=new Node;

        add->next=add\_next;

        add\_next->prev=add;

        add\_next->value=tmp;

        add\_next->next=nullptr;

        restored->size++;

    }

    restored->tail=add\_next;

    in.close();

    return restored;

}

int main() {

    DLinkedList \*list=createList();

    printList(&list);

    addToList(&list, 1, 10);

    addToList(&list, 2, 12);

    addToList(&list, 3, 31);

    addToList(&list, 4, 9);

    addToList(&list, 5, 200);

    addToList(&list, 6, -3);

    addToList(&list, 7, 55);

    printList(&list);

    addToList(&list, 1, deleteFromList(&list, 5));

    printList(&list);

    addToList(&list, 1, deleteFromList(&list, 5));

    printList(&list);

    char file[]="abc.txt";

    writeListToFile(&list, file);

    deleteList(&list);

    printList(&list);

    list = restoreListFromFile(file);

    printList(&list);

    deleteList(&list);

    return 0;

}

<https://github.com/artificial-intelligence-department/ai_programming_playground/pull/1036/files#diff-020ea50d7127efa142cfe22516e1afe93506eea4e2a9ee8f2132253fc5191d47>

Завдання №4 Algotester Lab 5

#include <iostream>

#include <vector>

using namespace std;

int main() {

    short rows, columns;

    cin >> rows >> columns;

    vector<vector<char>> cave(rows);

    for (short i = 0; i<rows; i++) {

        cave[i].resize(columns);

        for (short j=0; j<columns; j++) {cin >> cave[i][j];}

    }

    for (short j=0; j<columns; j++) {

        for (short i=rows-1; i>=0; i--) {

            if((cave[i][j] == 'S') && (i+1<rows) && (cave[i+1][j] == 'O')) {

                char tmp = cave[i][j];

                cave[i][j] = cave[i+1][j];

                cave[i+1][j] = tmp;

                i=rows-1;

            }

        }

    }

    for (short i=0; i<rows; i++) {

        for (short j=0; j<columns; j++) {

            cout << cave[i][j];

        }

        std::cout << '\n';

    }

    return 0;

}

<https://github.com/artificial-intelligence-department/ai_programming_playground/pull/1036/files#diff-48a7a07258d3919d6de5651c45008c7a7c7e88faba80af39e7d7ae26e878773d>

Завдання №5 Algotester Lab 7-8

#include <iostream>

#include <string>

#include <vector>

using namespace std;

class Dyn\_array {

private:

    long\* m\_array=NULL;

    long m\_size = 0;

    long m\_capacity = 1;

public:

    Dyn\_array() {}

    void insert(long index, long N, vector<long> to\_insert) {

        long new\_size;

        new\_size = m\_size+N;

        while (new\_size>=m\_capacity) { m\_capacity\*=2; }

        long \*new\_array = new long[m\_capacity];

        long i\_n=0, i\_o=0;

        while(i\_n<index) {

            new\_array[i\_n] = m\_array[i\_o];

            i\_n++, i\_o++;

        }

        for (long j=0; j<N; j++) {

            new\_array[i\_n] = to\_insert[j];

            i\_n++;

        }

        while(i\_n<new\_size) {

            new\_array[i\_n] = m\_array[i\_o];

            i\_n++, i\_o++;

        }

        delete[] m\_array;

        m\_size = new\_size;

        m\_array = new\_array;

    }

    void erase(long index, long N) {

        long new\_size;

        new\_size = m\_size-N;

        long \*new\_array = new long[m\_capacity];

        long i\_n=0, i\_o=0;

        while(i\_n<index) {

            new\_array[i\_n] = m\_array[i\_o];

            i\_n++, i\_o++;

        }

        while(N>0) { i\_o++, --N; }

        while(i\_n<new\_size) {

            new\_array[i\_n] = m\_array[i\_o];

            i\_n++, i\_o++;

        }

        delete[] m\_array;

        m\_size = new\_size;

        m\_array = new\_array;

    }

    long size() {

        return m\_size;

    }

    long capacity() {

        return m\_capacity;

    }

    long& operator[](long index);

    friend ostream& operator<<(ostream &out, Dyn\_array arr);

};

long& Dyn\_array::operator[](long index) {

    return m\_array[index];

}

ostream& operator<<(ostream &out, Dyn\_array arr) {

    for (long i=0; i<arr.m\_size; i++) {out << arr[i] << ' ';}

    return out;

}

int main() {

    Dyn\_array v;

    long Q;

    cin >> Q;

    vector<string> requests(Q);

    vector<long> change;

    vector<long> ins;

    for (long i=0; i<Q; i++) {

        cin >> requests[i];

        if (requests[i] == "") {i--;}

        else if(requests[i] == "insert") {

            long tmp;

            cin >> tmp;

            change.push\_back(tmp);

            cin >> tmp;

            change.push\_back(tmp);

            for (long j=0; j<tmp; j++) {

                long tmp2;

                cin >> tmp2;

                ins.push\_back(tmp2);

            }

        }

        else if(requests[i] == "erase") {

            long tmp;

            cin >> tmp;

            change.push\_back(tmp);

            cin >> tmp;

            change.push\_back(tmp);

        }

        else if(requests[i] == "get") {

            long tmp;

            cin >> tmp;

            change.push\_back(tmp);

        }

        else if(requests[i] == "set") {

            long tmp;

            cin >> tmp;

            change.push\_back(tmp);

            cin >> tmp;

            change.push\_back(tmp);

        }

    }

    long change\_counter = 0, ins\_counter = 0;

    for (long i = 0; i<Q; i++) {

        if (requests[i] == "insert") {

            vector<long> to\_push;

            for (long abc=0;abc<change[change\_counter+1];abc++) {to\_push.push\_back(ins[ins\_counter++]);}

            v.insert(change[change\_counter],change[change\_counter+1],to\_push);

            change\_counter+=2;

        }

        else if(requests[i] == "erase") {

            v.erase(change[change\_counter],change[change\_counter+1]);

            change\_counter+=2;

        }

        else if(requests[i] == "size") {

            cout << v.size() << endl;

        }

        else if(requests[i] == "capacity") {

            cout << v.capacity() << endl;

        }

        else if(requests[i] == "get") {

            cout << v[change[change\_counter++]] << endl;

        }

        else if(requests[i] == "set") {

            v[change[change\_counter]] = change[change\_counter+1];

            change\_counter+=2;

        }

        else if(requests[i] == "print") {

            cout << v << endl;

        }

    }

    return 0;

}

<https://github.com/artificial-intelligence-department/ai_programming_playground/pull/1036/files#diff-50753e9c4ec656a3e500ae5f4caa7c21df8f051848ac90922bb29ae6084ae5c0>

# Завдання №6.1 Class Practice Task 1-3

#include <iostream>

#include <fstream>

using namespace std;

struct Node {

    int value=0;

    struct Node \*next=nullptr;

    struct Node \*prev=nullptr;

};

struct DLinkedList {

    unsigned long size=0;

    Node \*head=nullptr;

    Node \*tail=nullptr;

};

DLinkedList\* createList() {

    DLinkedList \*tmp = new DLinkedList;

    tmp->size=0;

    tmp->head=nullptr, tmp->tail=nullptr;

    return tmp;

}

void printList(DLinkedList \*\*list) {

    if(!(\*list) || !(\*list)->size) {cout << "List is empty";}

    else {

        Node \*tmp=(\*list)->head;

        while (tmp) {

            cout << tmp->value << ' ';

            tmp=tmp->next;

        }

    }

    cout << '\n';

}

void addToList(DLinkedList \*\*list, double number, int value) {

    Node \*toAdd=new Node;

    toAdd->value=value;

    toAdd->next=nullptr;

    toAdd->prev=nullptr;

    if (!((\*list)->size)) {

        (\*list)->head=toAdd;

        (\*list)->tail=toAdd;

    }

    else if (number == 1) {

        Node \*next=(\*list)->head;

        toAdd->next=next;

        next->prev=toAdd;

        (\*list)->head=toAdd;

    }

    else if (number-1 == (\*list)->size) {

        Node \*prev=(\*list)->tail;

        toAdd->prev=prev;

        prev->next=toAdd;

        (\*list)->tail=toAdd;

    }

    else if ((\*list)->size) {

        Node \*prev=(\*list)->head;

        Node \*next=(\*list)->tail;

        for (unsigned long i=1;i<number-1;i++) {

            prev=prev->next;

        }

        next=prev->next;

        toAdd->prev=prev;

        toAdd->next=next;

        prev->next=toAdd;

        next->prev=toAdd;

    }

    (\*list)->size++;

}

int deleteFromList(DLinkedList \*\*list, double number) {

    int delVal;

    Node \*toDelete=(\*list)->head;

    if ((\*list)->size==1) {

        delVal=toDelete->value;

        delete (\*list)->head;

        (\*list)->head=nullptr;

        (\*list)->tail=nullptr;

    }

    else if (number == 1) {

        delVal=toDelete->value;

        Node \*next=(\*list)->head->next;

        delete toDelete;

        (\*list)->head=next;

    }

    else if (number == (\*list)->size) {

        toDelete=(\*list)->tail;

        delVal=toDelete->value;

        Node \*prev=(\*list)->tail->prev;

        delete toDelete;

        (\*list)->tail=prev;

    }

    else if ((\*list)->size) {

        for (unsigned long i=0;i<number-1;i++) {

            toDelete=toDelete->next;

        }

        delVal=toDelete->value;

        Node \*prev=toDelete->prev;

        Node \*next=toDelete->next;

        delete toDelete;

        prev->next=next;

        next->prev=prev;

    }

    (\*list)->size--;

    return delVal;

}

void deleteList(DLinkedList \*\*list) {

    Node \*tmp=(\*list)->head;

    if (tmp) { delete tmp; }

    (\*list)->size=0;

    delete (\*list);

    (\*list) = nullptr;

}

DLinkedList\* reverseList(DLinkedList \*\*original) {

    DLinkedList \*reversed=createList();

    if((\*original)->size) {

        Node \*tmp=(\*original)->tail;

        addToList(&reversed, 1, tmp->value);

        for (unsigned long i=2;i<=(\*original)->size;i++) {

            tmp=tmp->prev;

            addToList(&reversed, i, tmp->value);

        }

    }

    return reversed;

}

bool compareLists(DLinkedList \*\*first\_L, DLinkedList \*\*second\_L) {

    bool same=true;

    Node \*first=(\*first\_L)->head, \*second=(\*second\_L)->head;

    if ((\*first\_L)->size != (\*second\_L)->size) {same=false;}

    else {

        for (unsigned long i=0;i<(\*first\_L)->size;i++) {

            if (first->value != second->value) {same=false; break;}

            first=first->next, second=second->next;

        }

    }

    return same;

}

DLinkedList\* addLists(DLinkedList \*\*first\_L, DLinkedList \*\*second\_L) {

    DLinkedList \*result=createList();

    Node\* first=(\*first\_L)->head;

    Node\* second=(\*second\_L)->head;

    short carry=0;

    unsigned long index=1;

    while (first || second || carry) {

        int sum=(first!=nullptr?first->value:0)+(second!=nullptr?second->value:0)+carry;

        carry=sum/10;

        addToList(&result, index, sum%10);

        if(first!=nullptr)first=first->next;

        if(second!=nullptr)second=second->next;

        index++;

    }

    return result;

}

int main() {

    DLinkedList \*list=createList();

    printList(&list);

    addToList(&list, 1, 2);

    addToList(&list, 2, 3);

    addToList(&list, 3, 4);

    addToList(&list, 4, 5);

    addToList(&list, 5, 6);

    addToList(&list, 6, 7);

    addToList(&list, 7, 8);

    printList(&list);

    DLinkedList \*reversed=reverseList(&list);

    printList(&reversed);

    cout << (compareLists(&list, &reversed)?"true":"false") << endl;

    cout << (compareLists(&list, &list)?"true":"false") << endl;

    DLinkedList \*sum=addLists(&reversed, &list);

    printList(&sum);

    deleteList(&list), deleteList(&reversed), deleteList(&sum);

    return 0;

}

<https://github.com/artificial-intelligence-department/ai_programming_playground/pull/1036/files#diff-bb65ac6f8de47e7bb29340d72c64032ee90e5ee1e80fd35305f1ba1874747286>

Завдання №6.2 Class Practice Task 4-5

#include <iostream>

using namespace std;

struct TreeNode {

    int value=0;

    TreeNode\* left=nullptr;

    TreeNode\* right=nullptr;

};

TreeNode\* create\_mirror\_flip(TreeNode\* root) {

    if (!root) { return root; }

    TreeNode\* leaf=new TreeNode;

    leaf->value=root->value;

    leaf->left=create\_mirror\_flip(root->right);

    leaf->right=create\_mirror\_flip(root->left);

    return leaf;

}

void tree\_sum(TreeNode\* root) {

    if(root) {

        tree\_sum(root->left);

        tree\_sum(root->right);

        if ((!root->left) && (!root->right)) return;

        int left=((root->left!=nullptr)?root->left->value:0);

        int right=((root->right!=nullptr)?root->right->value:0);

        root->value=left+right;

    }

}

void print\_tree(TreeNode\* tree) {

    if (tree) {

        cout << tree->value << ' ';

        print\_tree(tree->left);

        print\_tree(tree->right);

    }

}

int main() {

    TreeNode\* tree=new TreeNode;

    tree->value=7;

    tree->left=new TreeNode;

    tree->left->value=6;

    tree->right=new TreeNode;

    tree->right->value=5;

    tree->left->left=new TreeNode;

    tree->left->left->value=4;

    tree->left->right=new TreeNode;

    tree->left->right->value=3;

    tree->right->left=new TreeNode;

    tree->right->left->value=2;

    tree->right->right=new TreeNode;

    tree->right->right->value=1;

    print\_tree(tree);

    cout << '\n';

    TreeNode\* mirroredTree=create\_mirror\_flip(tree);

    print\_tree(mirroredTree);

    cout << '\n';

    tree\_sum(tree);

    print\_tree(tree);

    cout << '\n';

    delete tree;

    delete mirroredTree;

    return 0;

}

<https://github.com/artificial-intelligence-department/ai_programming_playground/pull/1036/files#diff-a3f3508ad35d318bfba4363875d19e7b9a10e4afc921ba2fe0f524ff7c2e2469>

Завдання № 7.1 - Self Practice Task A

#include <iostream>

int main() {

    int n, k = 15;

    char c;

    std::cin >> n;

    for (int i = 0;i<n;i++) {

        std::cin >> c;

        switch (c) {

        case 'k':

            if (k > 0) {k-=1;}

            break;

        case 'p':

            if (k != 0) {k=15;}

            break;

        case 'z':

            if (k > 0) {k=0;}

            else {k = 15;}

            break;

        }

    }

    std::cout << k;

}

<https://github.com/artificial-intelligence-department/ai_programming_playground/pull/1036/files#diff-99866f62be64466cb3f195b6f9a06adb56cf06d1673aedc8d7560313190b8ecf>

Завдання № 7.2 - Self Practice Task 0921

#include <iostream>

int main() {

    long a, b;

    std::cin >> a >> b;

    if (a > b) {

        long c = a;

        a = b;

        b = c;

    }

    if (a+1 == b) std::cout << "-1";

    else std::cout << (b-1);

    return 0;

}

<https://github.com/artificial-intelligence-department/ai_programming_playground/pull/1036/files#diff-5753161ee47f21dcfba69fb5f661d30c10372f663376675f00959fffe30c539f>

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

Завдання №3 VNS Lab 10

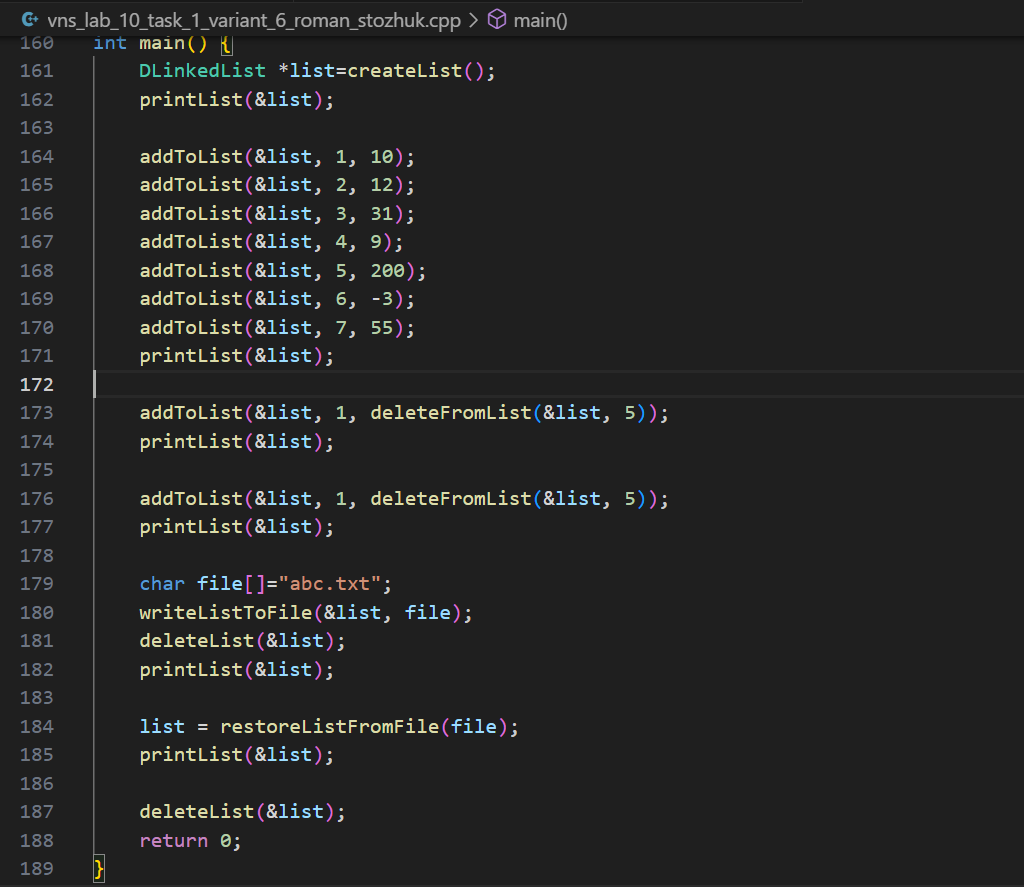


Рисунок Виклики функцій завдання №3

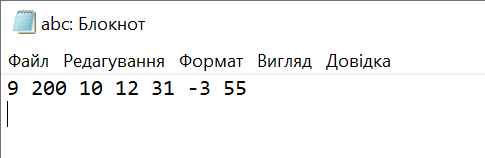


Рисунок Файл abc.txt після виконання завдання №3

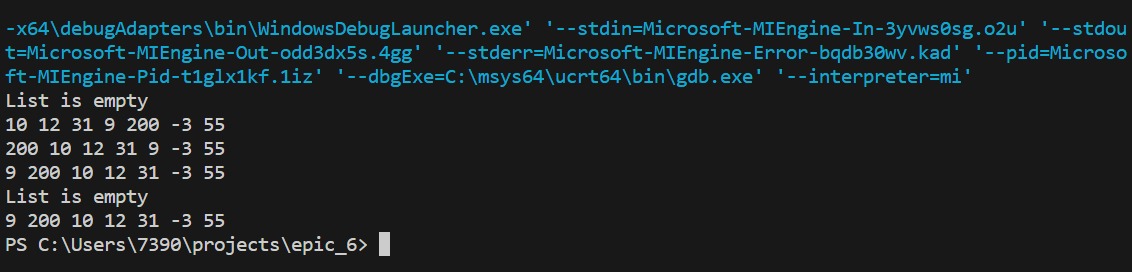


Рисунок Результат виконання завдання №3

Час затрачений на виконання завдання: 3год40хв

Завдання №4 Algotester Lab 5

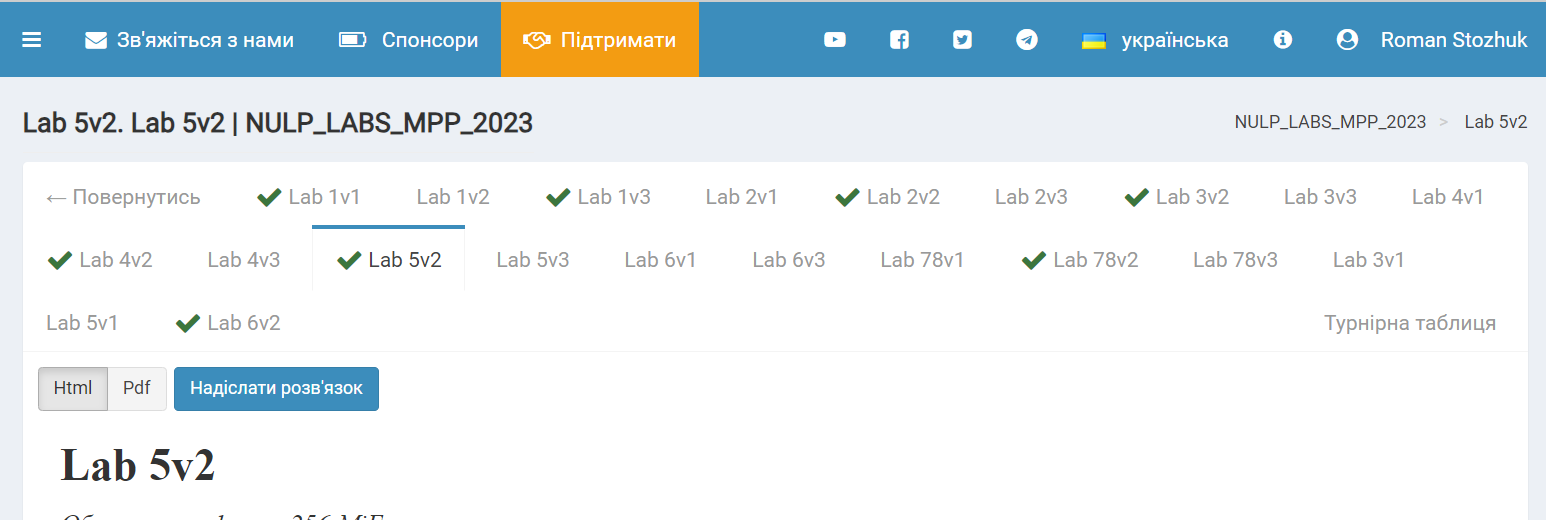


Рисунок Результат виконання завдання №4

Час затрачений на виконання завдання: 2год

Завдання №5 Algotester Lab 7-8

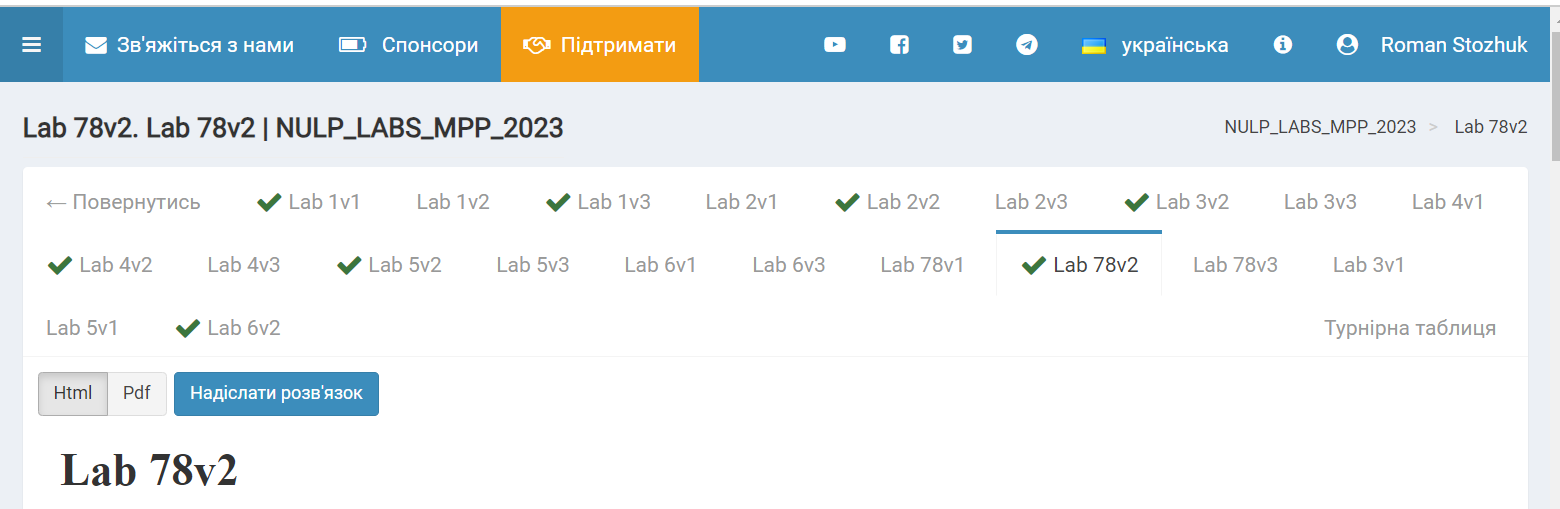


Рисунок Результат виконання завдання №5

Час затрачений на виконання завдання: 4год

# Завдання №6.1 Class Practice Task 1-3

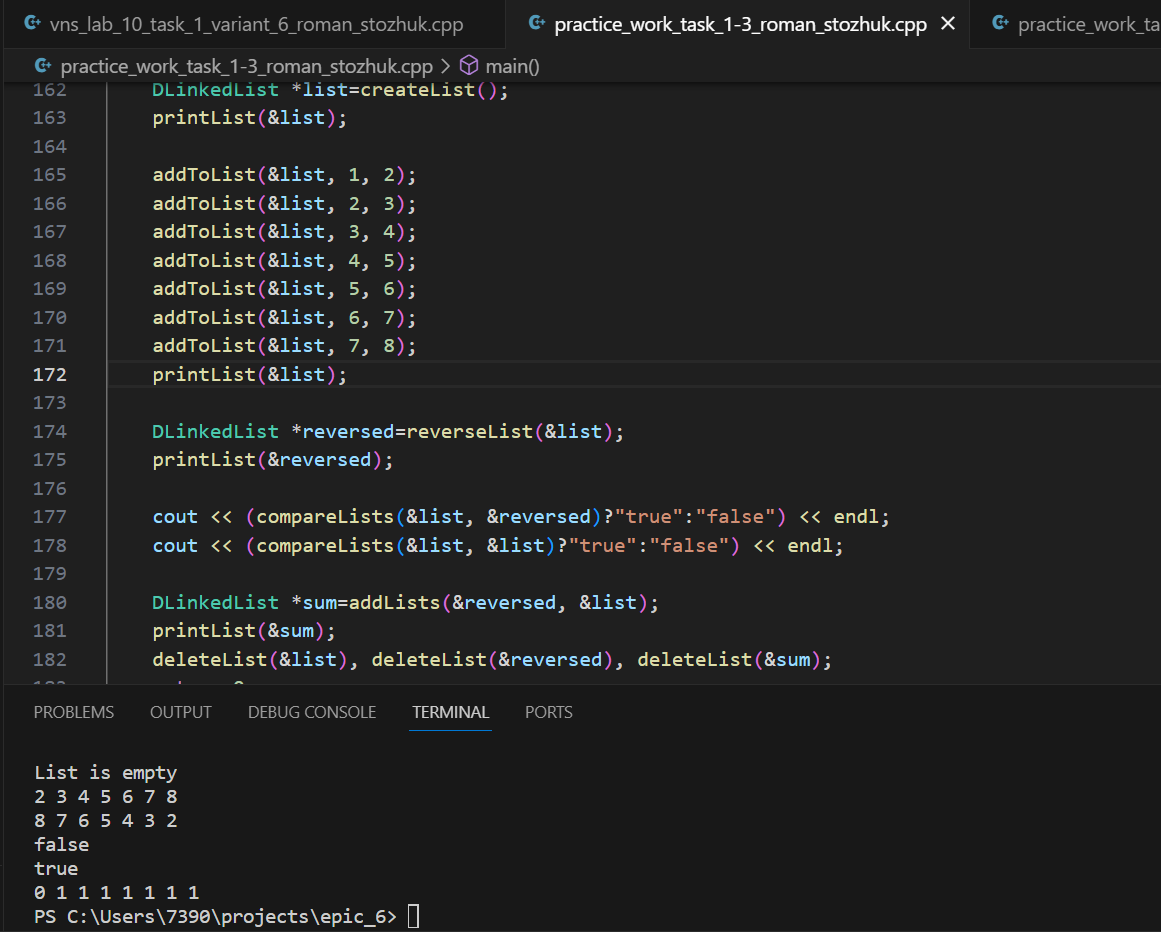


Рисунок Виклики функцій та результат виконання завдання №6.1

Час затрачений на виконання завдання: 2год

Завдання №6.2 Class Practice Task 4-5

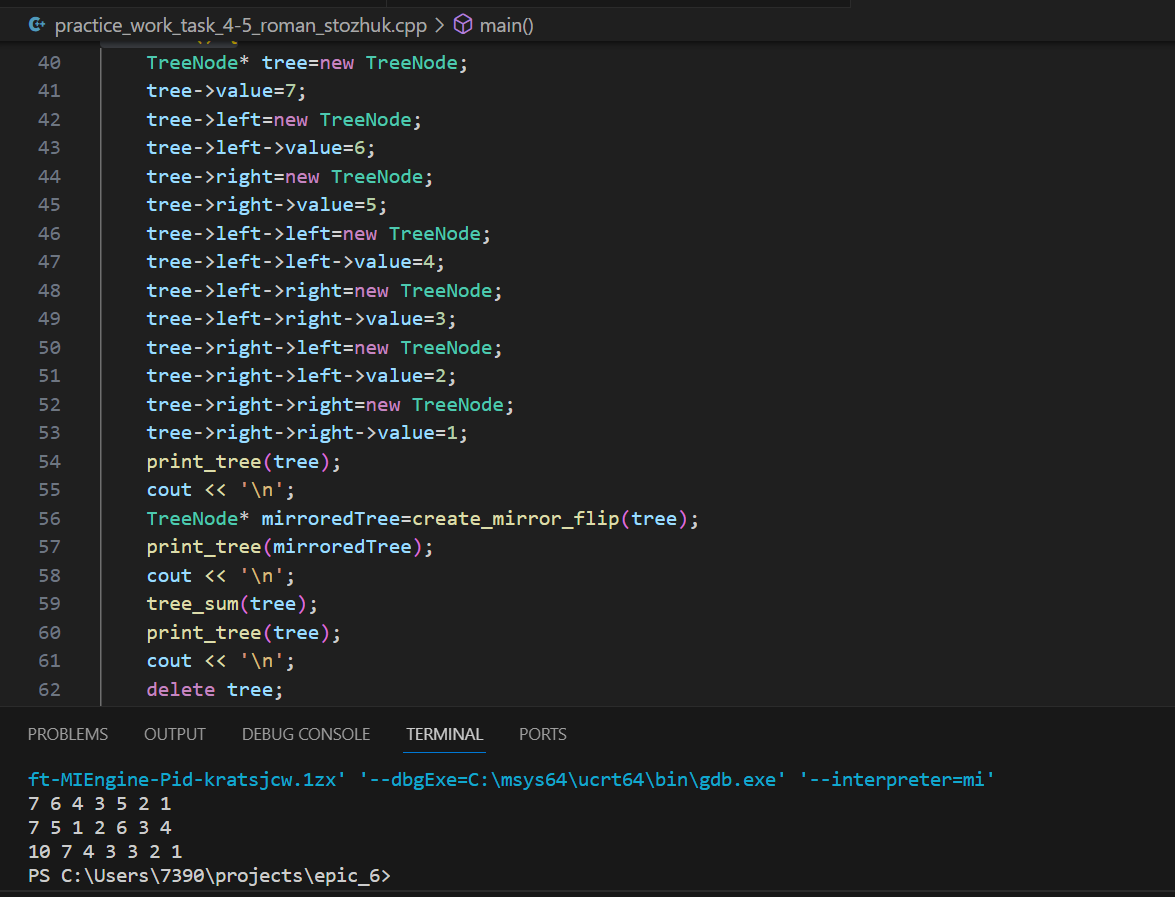


Рисунок Результат виконання завдання №6.2

Час затрачений на виконання завдання: 2год

Завдання № 7.1 - Self Practice Task A

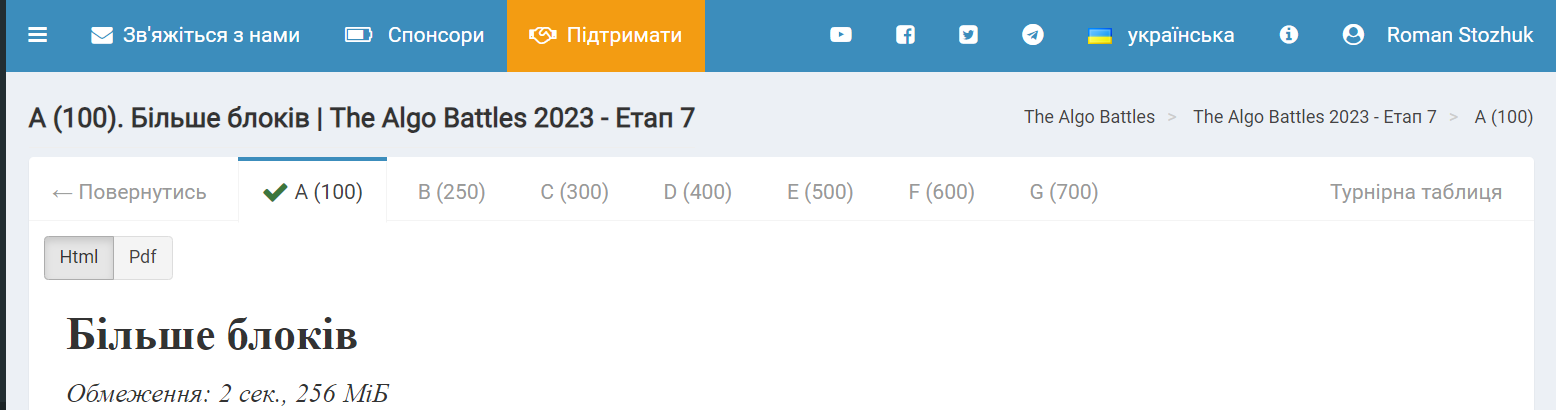


Рисунок Результат виконання завдання №7.1

Час затрачений на виконання завдання: 10хв

Завдання № 7.2 - Self Practice Task 0921

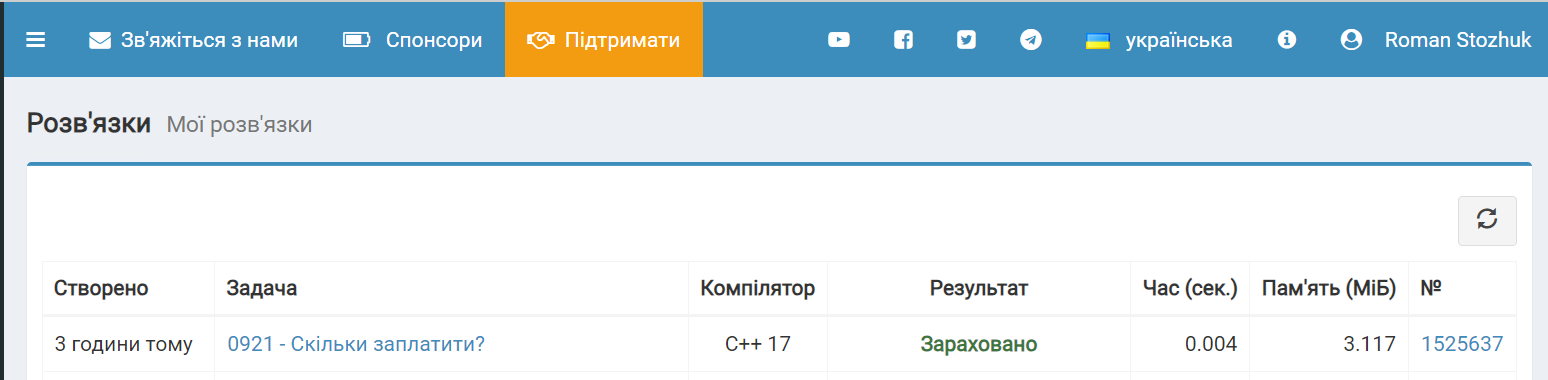
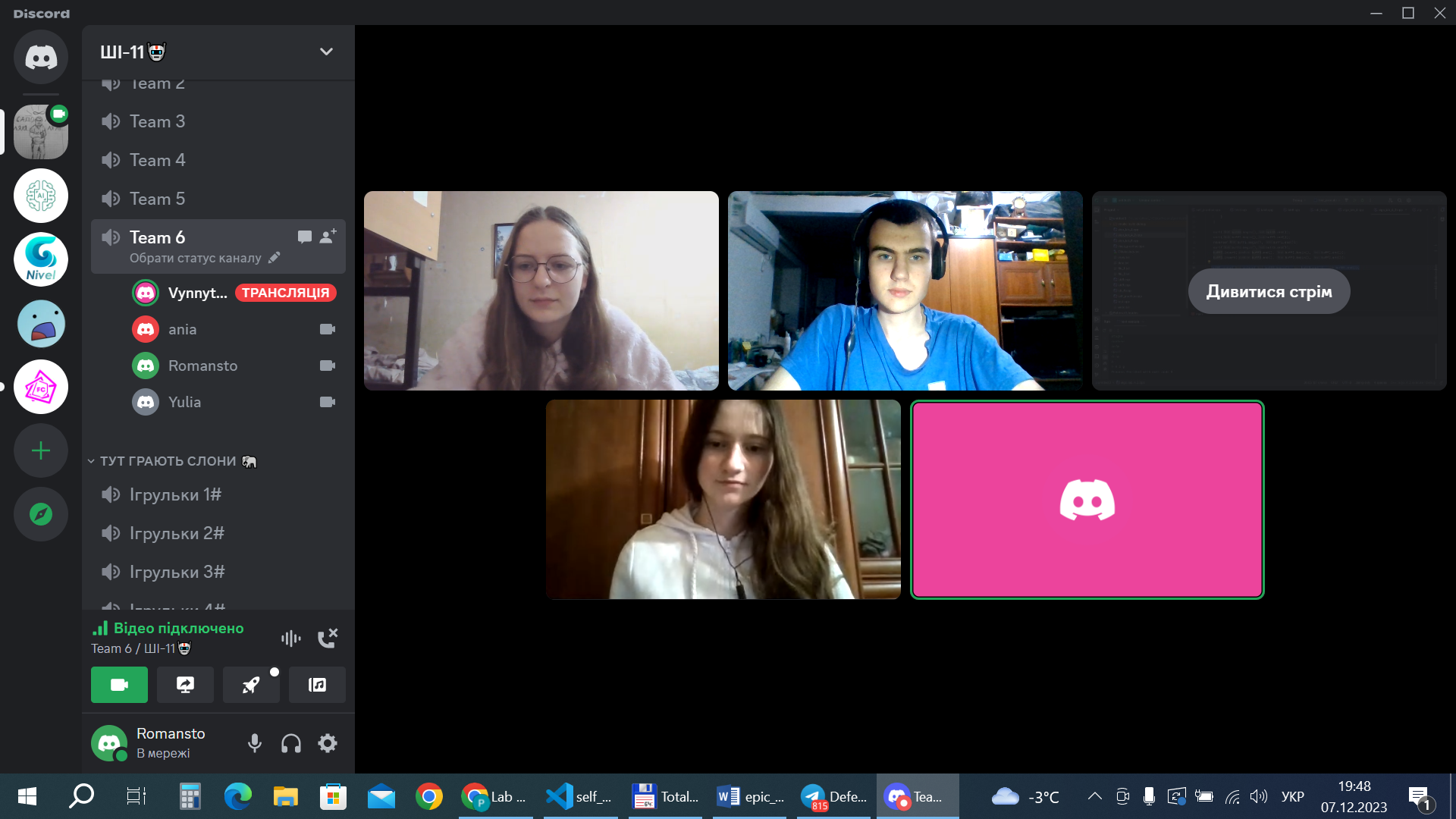


Рисунок Результат виконання завдання №7.2

Час затрачений на виконання завдання: 20хв

## **6. Кооперація з командою:**

* 

# **Висновки:**

Протягом шостого епіку я навчився працювати з динамічними структурами