Міністерство освіти і науки України

Національний університет «Львівська політехніка»

Кафедра систем штучного інтелекту



**Звіт**

про виконання

**Лабораторних та практичних робіт № 10**

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

***з розділу***: «\_\_\_\_\_\_\_\_Динамічні структури (Черга, Стек, Списки, Дерево). Алгоритми обробки динамічних структур \_\_\_\_\_\_\_\_\_\_»

***Виконав:***

студент групи ШІ-13

Кібиш Тарас Юрійович

# **Тема роботи:**

Вивчення базових динамічних структур. Робота з динамічними структурами. Алгоритми обробки динамічних структур.

# **Мета роботи:**

Вивчення основних понять про динамічні струкрури. Здобуття навичок в роботі з ними.

# **Теоретичні відомості:**

1. Теоретичні відомості з переліком важливих тем:

* Тема №1: Класи
* Тема №2: Робота з класами
* Тема №3: Однозвязні списки
* Тема №4: Двозвязні списки
* Тема №5\_: Бінарні дерева

1. Індивідуальний план опрацювання теорії:

* Тема №1: Класи
  + Джерела Інформації
    - <http://cpp.dp.ua/ponyattya-klasu/>
  + Що опрацьовано:
    - Збодуте розуміння, що таке класи
  + Статус: Ознайомлений
  + Початок опрацювання теми: 08.12.2023
  + Звершення опрацювання теми: 14.12.2023
* Тема №2: Робота з класами
  + Джерела Інформації:
    - <https://www.bestprog.net/uk/2018/03/25/c-classes-part-1-the-concept-of-class-declaring-a-data-type-class-the-object-of-the-class-classes-in-the-clr-encapsulation-of-data-in-a-class_ua/#q02>
  + Що опрацьовано:
    - Ознайомлений з основними методами роботи з класами
  + Статус: Ознайомлений частково
  + Початок опрацювання теми: 08.12.2023
  + Звершення опрацювання теми: 14.12.2023
* Тема №3: Однозвязні списки
  + Джерела Інформації:
    - <https://www.bestprog.net/uk/2022/02/11/c-linear-singly-linked-list-general-information-ua/>
    - <https://www.youtube.com/watch?v=-25REjF_atI&ab_channel=%D0%91%D0%BB%D0%BE%D0%B3%D0%B0%D0%BD%E2%A6%81%D0%A3%D1%80%D0%BE%D0%BA%D0%B8%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F>
  + Що опрацьовано:
    - Здобуте розуміння, що таке однозвязні списки
  + Статус: Ознайомлений
  + Початок опрацювання теми: 10.12.2023
  + Звершення опрацювання теми: 13.12.2023
* Тема №4: Двозвязні списки
  + Джерела Інформації:
    - <https://www.youtube.com/watch?v=QLzu2-_QFoE&t=734s&ab_channel=%D0%91%D0%BB%D0%BE%D0%B3%D0%B0%D0%BD%E2%A6%81%D0%A3%D1%80%D0%BE%D0%BA%D0%B8%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F>
  + Що опрацьовано:
    - Здобуто поняття, що таке двозвязні списки
  + Статус: Ознайомлений
  + Початок опрацювання теми: 09.12.2023
  + Звершення опрацювання теми: 14.12.2023
* Тема №5: Бінарні дерева.
  + Джерела Інформації:
    - <https://www.youtube.com/watch?v=QLzu2-_QFoE&t=734s&ab_channel=%D0%91%D0%BB%D0%BE%D0%B3%D0%B0%D0%BD%E2%A6%81%D0%A3%D1%80%D0%BE%D0%BA%D0%B8%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F>
  + Що опрацьовано:
    - Здобуто поняття, що таке бінарні дерева
  + Статус: Ознайомлений
  + Початок опрацювання теми: 10.12.2023
  + Звершення опрацювання теми: 15.12.2023

# **Виконання роботи:**

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

Завдання №1: Lab# programming: Algotester Lab 5

* Варіант завдання 3
* Деталі завдання
* У вас є карта гори розміром N×M.
* Також ви знаєте координати {x,y} , у яких знаходиться вершина гори.
* Ваше завдання - розмалювати карту таким чином, щоб найнижча точка мала число 0, а пік гори мав найбільше число.
* Клітинкі які мають суміжну сторону з вершиною мають висоту на один меншу, суміжні з ними і не розфарбовані мають ще на 1 меншу висоту і так далі.
* Важливі деталі для врахування в імплементації програми
* Обмеження

1≤N,M≤10

1≤x≤N

1≤y≤M

Завдання №2:  Lab# programming: Algotester Lab 7-8

* Варіант завдання 2
* Деталі завдання

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 перегрузку оператора [][]
* **Модифікація значення** i-го елементу  
  Ідентифікатор - set  
  Ви отримуєте 2 цілих числа - індекс елемента, який треба змінити, та його нове значення. Реалізувати використовуючи перегрузку оператора [][]
* **Вивід динамічного масиву на екран**  
  Ідентифікатор - print  
  Ви не отримуєте аргументів.  
  Ви виводите усі елементи динамічного масиву через пробіл.  
  Реалізувати використовуючи перегрузку оператора <<
* Важливі деталі для врахування в імплементації програми
* Обмеження

0≤Q≤105  
0≤li≤105  
∥l∥≤105

Завдання №3: Lab# programming: VNS Lab 10

* Варіант завдання 12
* Деталі завдання

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.

12.Записи в лінійному списку містять ключове поле типу \*char (рядок

символів). Сформувати двонаправлений список. Знищити з нього елементи,

з однаковими ключовими полями. Додати елемент після елемента із заданим

ключовим полем.  
Завдання №4: Practice# programming:  Self Practice Task

* Варіант завдання
* Деталі завдання

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)(−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≥1, і сигнал силою 0, якщо k=0=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≥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 №5: Practice# programming: Class Practice Task

* Варіант завдання
* Деталі завдання

## Задача №1 - 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;

## Задача №2 - 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***.

## Задача №3 – 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.

## Задача №4 - 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

* Важливі деталі для врахування в імплементації програми

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

Програма №2 Lab# programming: Algotester Lab 7-8

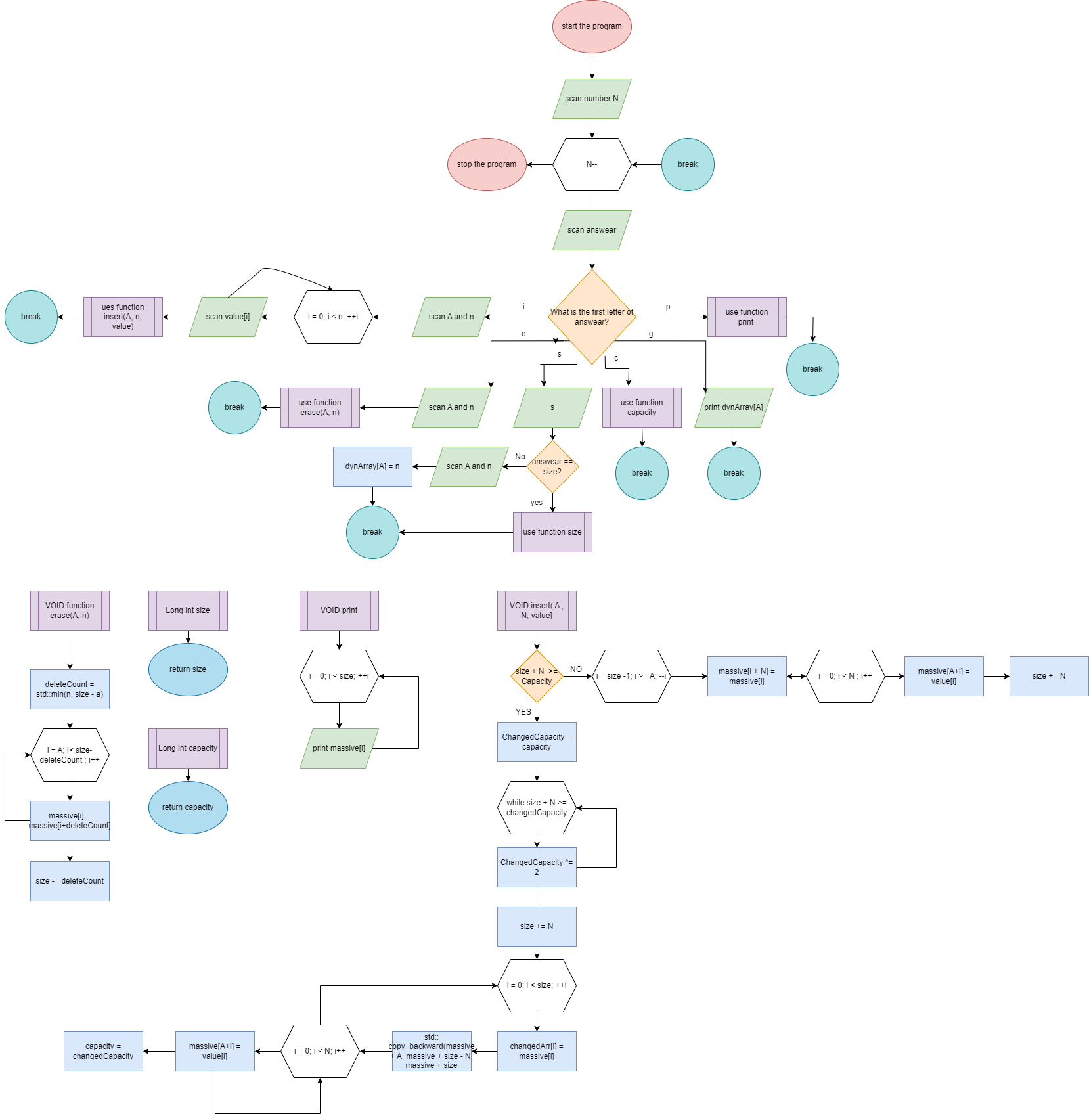
* Блок-схема
* 

Рисунок 1: Lab# programming: Algotester Lab 7-8

* Планований час на реалізацію 5 годин
* Важливі деталі для врахування в імплементації

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

Завдання №\_\_ Деталі по конфігурації середовища + скріншоти з підписами до скріншотів.

Підпис та № до блоку з скріншотами до конфігурації

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

Завдання №1: Lab# programming: Algotester Lab 5

#include <iostream>

#include <vector>

using *namespace* std;

*void* up(*int* *N*, *int* *M*, vector<vector<*int*>> &*massive*, *int* *x*, *int* *y*) {

    if (*x* == 0) {

        return;

    }

    for (*int* i = 0; i < *M*; i++) {

*massive*[*x* - 1][i] = *massive*[*x*][i] - 1;

    }

        up(*N*, *M*, *massive*, *x* - 1, *y*);

}

*void* down(*int* *N*, *int* *M*, vector<vector<*int*>> &*massive*, *int* *x*, *int* *y*) {

     if (*x*  == *N*-1) {

        return;

    }

    for (*int* i = 0; i < *M*; i++) {

*massive*[*x*+1][i] = *massive*[*x*][i] - 1;

    }

        down(*N*, *M*, *massive*, *x* + 1, *y*);

}

*int* main() {

*int* N, M;

*int* x, y;

    cin >> N >> M;

    cin >> x >> y;

    vector<vector<*int*>> field(N, vector<*int*>(M, 0));

*int* start;

*int* x1 = x, y1 = y;

    if(x\*2 <= N+1 &&  y\*2 <= M+1){

         start = (N - x) + (M - y);

    }

    else{

        if(x >= N/2 + 1){

            x1 = N -x +1;

        }

        if( y >= M/2 +1){

            y1 = M -y +1;

        }

        start = (N - x1) + (M - y1);

    }

    x--;

    y--;

    field[x][y] = start;

*int* t = 1;

    for (*int* i = y + 1; i < M; i++) {

        field[x][i] = start - t;

        t++;

    }

    t = 1;

    for (*int* i = y - 1; i >= 0; i--) {

        field[x][i] = start - t;

        t++;

    }

if(N > 1){

    up(N, M, field, x, y);

    down(N, M, field, x, y);

     for (*int* i = 0; i < N; ++i) {

        for (*int* j = 0; j < M; ++j) {

            cout << field[i][j] << " ";

        }

        cout << endl;

    }

}

else{

    for (*int* i = 0; i < N; ++i) {

        for (*int* j = 0; j < M; ++j) {

            cout << field[i][j] << " ";

        }

        cout << endl;

    }

}

    return 0;

}

Підпис та № до блоку з кодом програми

Завдання №2 Lab# programming: Algotester Lab 7-8

#include <iostream>

#include <string>

#include <algorithm>

*class* DynamicArray {

*private:*

*long* *int*\* massive;

*long* *int* size;

*long* *int* capacity;

*public:*

    DynamicArray() : size(0), capacity(1) {

         massive = new *long* *int*[capacity];

    }

    ~DynamicArray() {

        delete[] massive;

    }

*void* insert(*long* *int* *A*, *long* *int* *N*, *long* *int*\* *value*) {

        if (*A* < 0 || *A* > size)

            return;

        if (size + *N* >= capacity) {

*long* *int* changedCapacity = capacity;

            while (size + *N* >= changedCapacity) {

                changedCapacity \*= 2;

            }

*long* *int*\* changedArr = new *long* *int*[changedCapacity];

            size += *N*;

            for ( *int* i = 0; i < size; ++i) {

                changedArr[i] = massive[i];

            }

             delete[] massive;

            massive = changedArr;

            std::copy\_backward(massive + *A*, massive + size - *N*, massive + size);

            for (size\_t i = 0; i < *N*; ++i) {

                massive[*A* + i] = *value*[i];

            }

            capacity = changedCapacity;

        } else {

            for ( *int* i = size - 1; i >= *A*; --i) {

                massive[i + *N*] = massive[i];

            }

            for ( *int* i = 0; i < *N*; ++i) {

                massive[*A* + i] = *value*[i];

            }

            size += *N*;

        }

    }

*void* erase(*long* *int* *A*, *long* *int* *n*) {

*long* *int* deleteCount = std::min(*n*, size - *A*);

        for ( *int* i = *A*; i < size - deleteCount; ++i) {

            massive[i] = massive[i + deleteCount];

        }

        size -= deleteCount;

    }

*long* *int* Size() {

        return size;

    }

*long* *int* Capacity() {

        return capacity;

    }

*void* print(){

        for( *int* i = 0; i < size; ++i){

            std::cout << massive[i] << " ";

        }

    }

*long* *int*& operator[](*long* *int* *A*) {

        return massive[*A*];

    }

};

*int* main() {

    using *namespace* std;

*long* *int* N;

    cin >> N;

    DynamicArray dynArray;

    while (N--) {

        string answer;

        cin >> answer;

*char* sym = answer[0];

*long* *int* A, n;

*long* *int*\* value;

        switch (sym) {

            case 'i':

                cin >> A >> n;

                value = new *long* *int*[n];

                for ( *int* i = 0; i < n; ++i) {

                    cin >> value[i];

                }

                dynArray.insert(A, n, value);

                delete[] value;

                break;

            case 'e':

                cin >> A >> n;

                dynArray.erase(A, n);

                break;

            case 's':

                if (answer == "size") {

                    cout << dynArray.Size() << endl;

                } else if (answer == "set") {

                    cin >> A >> n;

                    if (A >= 0 && A < dynArray.Size()) {

                        dynArray[A] = n;

                    }

                }

                break;

            case 'c':

                cout << dynArray.Capacity() << endl;

                break;

            case 'g':

                cin >> A;

                if (A >= 0 && A < dynArray.Size()) {

                    cout << dynArray[A] << endl;

                }

                break;

            case 'p':

                dynArray.print();

                break;

            default:

                cout << "Unknown action\n";

        }

    }

    return 0;

}

Підпис та № до блоку з кодом програми

Завдання №3 Lab# programming: VNS Lab 10

#include <iostream>

#include <fstream>

#include <cstring>

*struct* Node {

*char*\* key;

    Node\* next;

    Node\* prev;

};

*class* LinkedList {

*public:*

    LinkedList() : head(nullptr), tail(nullptr) {}

    ~LinkedList() {

        destroyList();

    }

*void* createList() {

        head = tail = nullptr;

    }

*void* addElement(const *char*\* *newKey*) {

        Node\* newNode = new Node;

        newNode->key = strdup(*newKey*);

        newNode->next = nullptr;

        if (!head) {

            head = tail = newNode;

            newNode->prev = nullptr;

        } else {

            tail->next = newNode;

            newNode->prev = tail;

            tail = newNode;

        }

    }

*void* deleteElement(const *char*\* *keyToDelete*) {

        Node\* current = head;

        while (current) {

            if (strcmp(current->key, *keyToDelete*) == 0) {

                if (current->prev) {

                    current->prev->next = current->next;

                } else {

                    head = current->next;

                }

                if (current->next) {

                    current->next->prev = current->prev;

                } else {

                    tail = current->prev;

                }

                delete[] current->key;

                delete current;

                return;

            }

            current = current->next;

        }

    }

*void* printList() {

        Node\* current = head;

        while (current) {

            std::cout << current->key << " ";

            current = current->next;

        }

        std::cout << std::endl;

    }

*void* writeToFile(const *char*\* *filename*) {

        std::ofstream outFile(*filename*);

        Node\* current = head;

        while (current) {

            outFile << current->key << std::endl;

            current = current->next;

        }

        outFile.close();

    }

*void* destroyList() {

        Node\* current = head;

        while (current) {

            Node\* next = current->next;

            delete[] current->key;

            delete current;

            current = next;

        }

        head = tail = nullptr;

    }

*void* restoreFromFile(const *char*\* *filename*) {

        std::ifstream inFile(*filename*);

        if (!inFile.is\_open()) {

            std::cerr << "Error opening file: " << *filename* << std::endl;

            return;

        }

*char* buffer[256];

        while (inFile.getline(buffer, sizeof(buffer))) {

            addElement(buffer);

        }

        inFile.close();

    }

*void* deleteElementsByKey(const *char*\* *keyToDelete*) {

    Node\* current = head;

    while (current) {

        if (strcmp(current->key, *keyToDelete*) == 0) {

            Node\* next = current->next;

            Node\* prev = current->prev;

            if (prev) {

                prev->next = next;

            } else {

                head = next;

            }

            if (next) {

                next->prev = prev;

            } else {

                tail = prev;

            }

            delete[] current->key;

            delete current;

            current = next;

        } else {

            current = current->next;

        }

        }

    }

*void* addElementAfterKey(const *char*\* *keyToFind*, const *char*\* *newKey*) {

        Node\* current = head;

        while (current) {

            if (strcmp(current->key, *keyToFind*) == 0) {

                Node\* newNode = new Node;

                newNode->key = strdup(*newKey*);

                newNode->next = current->next;

                newNode->prev = current;

                if (current->next) {

                    current->next->prev = newNode;

                } else {

                    tail = newNode;

                }

                current->next = newNode;

                return;

            }

            current = current->next;

        }

    }

*private:*

    Node\* head;

    Node\* tail;

};

*int* main() {

    LinkedList list;

    list.createList();

    list.addElement("One");

    list.addElement("Two");

    list.addElement("Three");

     list.addElement("Two");

    std::cout << "Початковий лист: ";

    list.printList();

    list.writeToFile("output.txt");

    LinkedList newList;

    newList.restoreFromFile("output.txt");

    newList.deleteElementsByKey("Two");

    std::cout << "лист після видалення 'Two': ";

    newList.printList();

    newList.addElementAfterKey("One", "Yuxu");

    std::cout << "лист з добавленим новим елементом: ";

    newList.printList();

    return 0;

}

Підпис та № до блоку з кодом програми

Завдання №4: Practice# programming:  Self Practice Task

#include <iostream>

*int* main(){

    using *namespace* std;

*int* n;

*unsigned* *int* in = 15;

*char* ans;

cin >> n;

for(*int* i = 0; i < n; i++){

cin >> ans;

if(ans == 'k'){

if(in > 0)

in--;

}

else if(ans == 'p'){

    if(in>0)

    in = 15;

    else{

        in = 0;

    }

}

else if(ans == 'z'){

    if(in>0)

    in = 0;

    else{

    in = 15;

    }

}

}

cout << in;

    return 0;

}

Підпис та № до блоку з кодом програми

Завдання №5(task 1): Practice# programming: Class Practice Task

#include <iostream>

*struct* Node {

*int* data;

    Node\* next;

    Node(*int* *value*) : data(*value*), next(nullptr) {}

};

Node\* reverse(Node\* *head*) {

    Node\* prev = nullptr;

    Node\* current = *head*;

    Node\* next = nullptr;

    while (current != nullptr) {

        next = current->next;

        current->next = prev;

        prev = current;

        current = next;

    }

*head* = prev;

    return *head*;

}

*void* printList(Node\* *node*) {

    while (*node* != nullptr) {

        std::cout << *node*->data << " ";

*node* = *node*->next;

    }

    std::cout << std::endl;

}

*int* main() {

    Node\* head = new Node(1);

    head->next = new Node(2);

    head->next->next = new Node(3);

    head->next->next->next = new Node(4);

    head->next->next->next->next = new Node(5);

    std::cout << "Оригінальний список: ";

    printList(head);

    head = reverse(head);

    std::cout << "Реверсивний список: ";

    printList(head);

    return 0;

}

Підпис та № до блоку з кодом програми

Завдання №5(task 2): Practice# programming: Class Practice Task

#include <iostream>

*struct* Node {

*int* data;

    Node\* next;

    Node(*int* *value*) : data(*value*), next(nullptr) {}

};

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

    while (*h1* != nullptr && *h2* != nullptr) {

        if (*h1*->data != *h2*->data) {

            return false;

        }

*h1* = *h1*->next;

*h2* = *h2*->next;

    }

    return (*h1* == nullptr && *h2* == nullptr);

}

*int* main() {

    Node\* head1 = new Node(1);

    head1->next = new Node(2);

    head1->next->next = new Node(3);

    Node\* head2 = new Node(1);

    head2->next = new Node(2);

    head2->next->next = new Node(3);

*bool* result = compare(head1, head2);

    if (result) {

        std::cout << "Списки ідентичні." << std::endl;

    } else {

        std::cout << "Списки різні." << std::endl;

    }

    return 0;

}

Підпис та № до блоку з кодом програми

Завдання №5(task 3): Practice# programming: Class Practice Task

#include <iostream>

*struct* Node {

*int* data;

    Node\* next;

    Node(*int* *value*) : data(*value*), next(nullptr) {}

};

*int* add(Node\* *n1*){

*int* r = 1;

*int* result = 0;

    while(*n1* != nullptr){

       result += (*n1*->data) \* r;

       r\*= 10;

*n1* = *n1* -> next;

    }

    return result;

}

*int* main() {

    Node\* number1 = new Node(4);

    number1->next = new Node(8);

    number1->next->next = new Node(9);

*int* result = add(number1);

   std::cout << result;

    return 0;

}

Підпис та № до блоку з кодом програми

Завдання №5(task 4): Practice# programming: Class Practice Task

#include <iostream>

*struct* TreeNode {

*int* val;

    TreeNode \*left;

    TreeNode \*right;

    TreeNode(*int* *x*) : val(*x*), left(NULL), right(NULL) {}

     ~TreeNode() {

    delete left;

    delete right;

     }

};

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

    if (!*root*)

     return nullptr;

   TreeNode \*mirroredRoot = new TreeNode(*root*->val);

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

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

    return mirroredRoot;

}

// Допоміжна функція для виводу дерева в порядку обходу в ширину

*void* printTree(TreeNode \**root*) {

    if (*root* == NULL) {

        return;

    }

    std::cout << *root*->val << " ";

    printTree(*root*->left);

    printTree(*root*->right);

}

*int* main() {

    // Приклад використання

    TreeNode \*root = new TreeNode(7);

    root->left = new TreeNode(5);

    root->right = new TreeNode(8);

    root->left->left = new TreeNode(4);

    root->right->right = new TreeNode(9);

    std::cout << "Original tree: ";

    printTree(root);

    std::cout << std::endl;

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

    std::cout << "Mirrored tree: ";

    printTree(mirrorRoot);

    std::cout << std::endl;

    return 0;

}

Підпис та № до блоку з кодом програми

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

Завдання №1: Lab# programming: Algotester Lab 5

3 9

1 2

8 9 8 7 6 5 4 3 2

7 8 7 6 5 4 3 2 1

6 7 6 5 4 3 2 1 0

PS E:\coding\Practise\epic 6\class\_practise>

Час затрачений на виконання завдання 8 годин

Завдання №2:  Lab# programming: Algotester Lab 7-8

12

size

0

capacity

1

insert 0 2

100 100

size

2

capacity

4

insert 0 2

102 102

size

4

capacity

8

insert 0 2

103 103

size

6

capacity

8

print

103 103 102 102 100 100

PS E:\coding\Practise\epic 6\class\_practise>

Час затрачений на виконання завдання 12 годин

Завдання №3: Lab# programming: VNS Lab 10

Початковий лист: One Two Three Two

лист після видалення 'Two': One Three

лист з добавленим новим елементом: One Yuxu Three

PS E:\coding\Practise\epic 6>

Час затрачений на виконання завдання 4 години

Завдання №4: Practice# programming:  Self Practice Task

5

pkkpk

14

PS E:\coding\Practise\epic 6\class\_practise>

Час затрачений на виконання завдання 40 хвилин

Завдання №5(task 1): Practice# programming: Class Practice Task

Оригінальний список: 1 2 3 4 5

Реверсивний список: 5 4 3 2 1

PS E:\coding\Practise\epic 6\class\_practise>

Час затрачений на виконання завдання 1 година

Завдання №5(task 2): Practice# programming: Class Practice Task

Списки ідентичні.

PS E:\coding\Practise\epic 6\class\_practise>

Час затрачений на виконання завдання 2 години

Завдання №5(task 3): Practice# programming: Class Practice Task

984

PS E:\coding\Practise\epic 6\class\_practise>

Час затрачений на виконання завдання 2 години

Завдання №5(task 4): Practice# programming: Class Practice Task

Original tree: 7 5 4 8 9

Mirrored tree: 7 8 9 5 4

PS E:\coding\Practise\epic 6\class\_practise>

Час затрачений на виконання завдання 1 година

## **6. Кооперація з командою:**

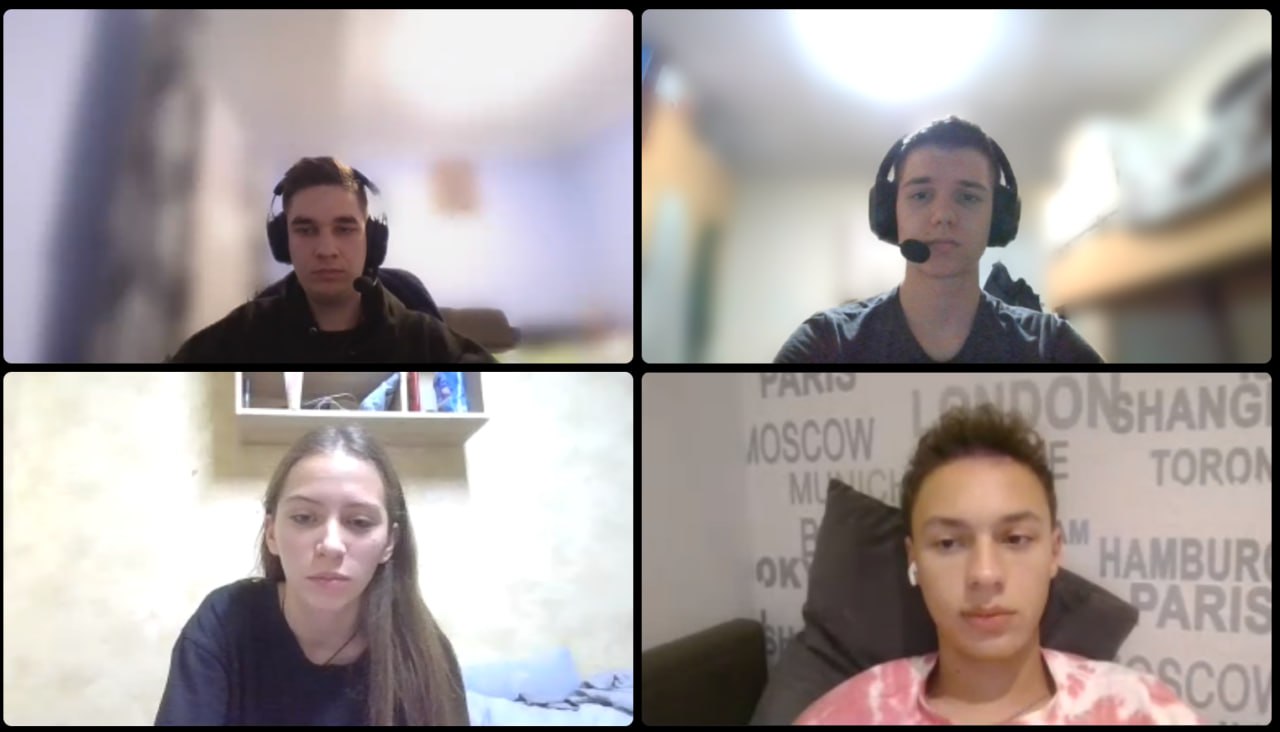
* Скрін з 1-ї зустрічі по обговоренню задач Епіку та Скрін прогресу по Трелло

Рисунок : зустріч з командою

* Скрін з 2-ї зустрічі по обговоренню задач Епіку та Скрін прогресу по Трелло
* Скрін з 3-ї зустрічі по обговоренню задач Епіку та Скрін прогресу по Трелло (опційно)
* Скрін з 2-му коментарями від учасників команди на пул реквесті з Ревю Роботи

# **Висновки:**

Під час виконання даної роботи я дізнався, що таке таке динамічні структури, які є їхні види, і для чого вони використовуються. Для цього я створив елементарні програми для виконання певних алгоритмів, з їхнім використанням,