Міністерство освіти і науки України

Національний університет «Львівська політехніка»

Кафедра систем штучного інтелекту

A blue and white logo

Description automatically generated

**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-13

Орза Євгеній Сергійович

# **Тема роботи:**

# Динамічні структури,Обробка дерев.

# **Мета роботи:**

Вивчення тереоретичних відомостей про структуру бінарнех дерев та динамічних структур.

**Теоретичні відомості:**

1. Теоретичні відомості з переліком важливих тем:

* Тема №1:Динамічні структури
* Тема №2: Обробка дерев

1. Індивідуальний план опрацювання теорії:

* Тема №1: Динамічні структури
  + Джерела Інформації:
* ЧатGPT.
* <https://www.bestprog.net/uk/2023/08/01/c-binary-search-tree-binarytree-class-integers-ua/>
* <https://allref.com.ua/uk/skachaty/Dinamichni_strukturi_danih_%28S++%29>

Що опрацьовано:

Реалізовані динамічні структури,

* + Статус: частково Ознайомлений.
  + Початок опрацювання теми:10/12
  + Звершення опрацювання теми: 12/12
* Тема №2 Дерева
* Джерела Інформації
* <https://www.youtube.com/watch?v=GzJoqJO1zdI&ab_channel=AaronJack>
* <https://www.youtube.com/watch?v=UHxtjVsOTHc&ab_channel=CodeBeauty>
* <https://www.youtube.com/watch?v=COZK7NATh4k&ab_channel=mycodeschool>
* <https://www.youtube.com/watch?v=fAAZixBzIAI&ab_channel=freeCodeCamp.org>

Що опрацьовано:

Здобувши практичний досвід, я реалізував операції вставки нових вузлів, видалення елементів та ефективний обхід дерева. Розглядаючи конкретні завдання, я вдосконалив свої навички у реалізації балансування дерева, пошуку мінімального та максимального значень, а також вивчив оптимізації для підвищення продуктивності операцій з деревами. Цей досвід сприяв моєму кращому розумінню структур даних і алгоритмів у контексті реальних завдань програмування.

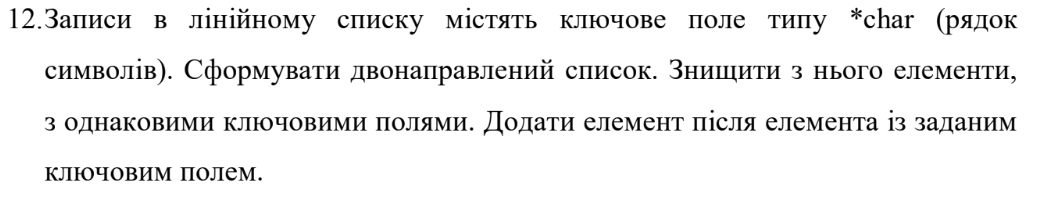
* + Статус: частково Ознайомлений
  + Початок опрацювання теми: 10/12
  + Звершення опрацювання теми: 12/12

# **Виконання роботи:**

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

VNS Lab 10

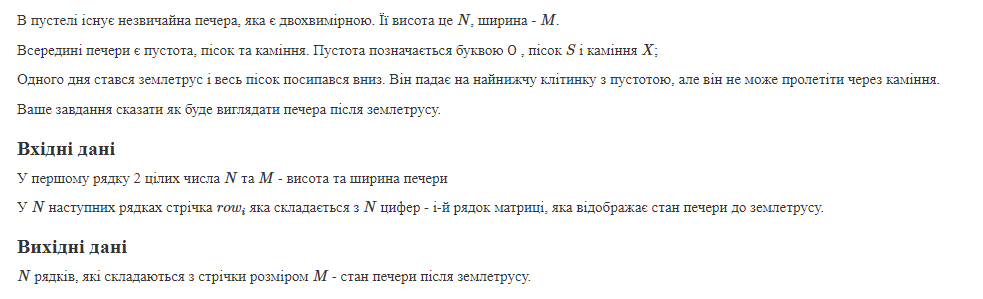
* Варіант 12



Рисунок

Algotester Lab 5 task 2

* Варіант 2



Рисунок

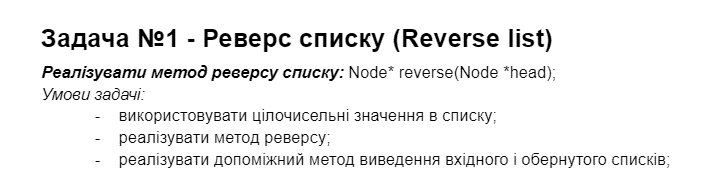
Algotester Lab 7\_8 task 2

* Варіант 2

Зображення, що містить текст, документ, знімок екрана, Шрифт

Автоматично згенерований опис

Рисунок



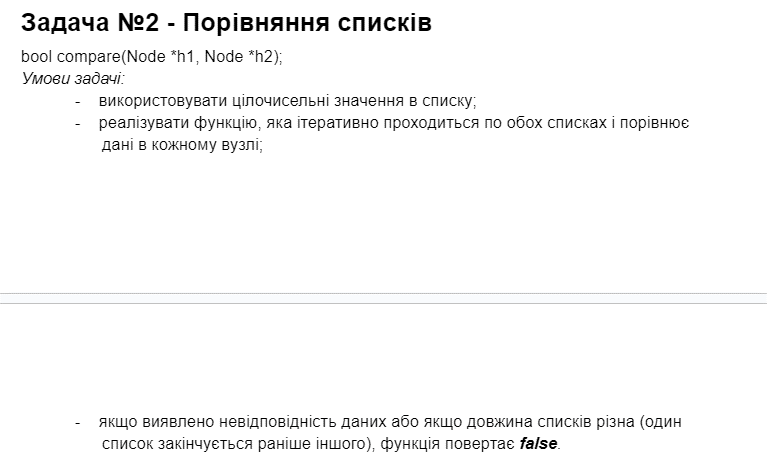


Рисунок Class Practice Task

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

VNS Lab 10

* Планований час - 2 години

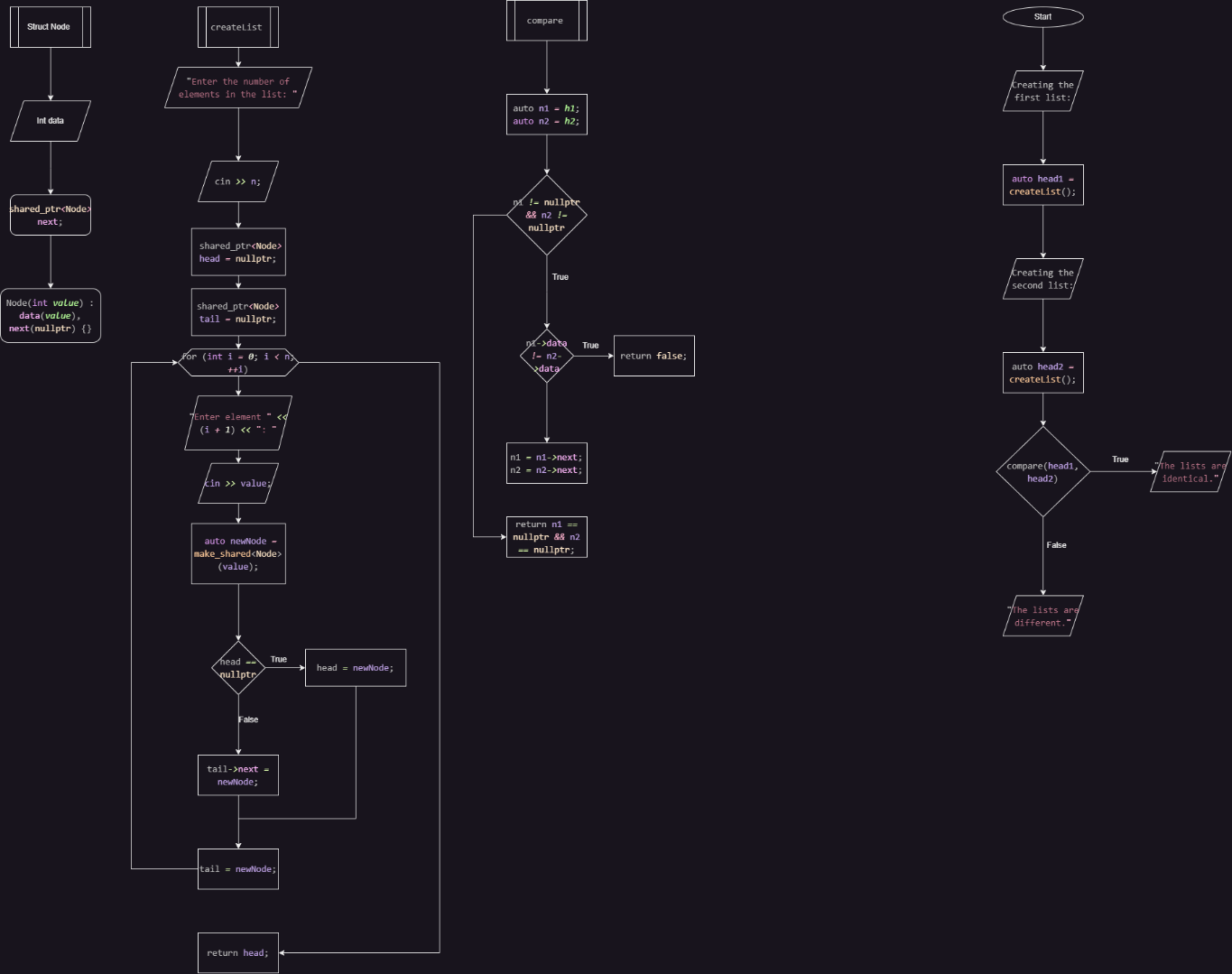
Algotester Lab 5 task 2

* Планований час - 30 хвилин

Algotester Lab 7\_8 task 2

* Планований час - 4 години

Class Practice Task

* Планований час - 4 години  
  

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

**Завдання 1:** VNS Lab 10. <https://github.com/artificial-intelligence-department/ai_programming_playground/pull/905/files#diff-01fa344e476a69740c2074453e4b1f2df879b23686940bd177b02dee9802b9d1>

*#include* <iostream>

*#include* <memory>

*#include* <string>

*#include* <unordered\_set>

*using* *namespace* std;

struct Node {

    string key;

    shared\_ptr*<*Node*>* next;

    weak\_ptr*<*Node*>* prev;

*Node*(const string& *value*) : key(*value*), next(nullptr) {}

};

shared\_ptr<Node> createList() {

    cout *<<* "Enter the number of elements in the list: ";

    int n;

    cin *>>* n;

    shared\_ptr*<*Node*>* head *=* nullptr;

    shared\_ptr*<*Node*>* tail *=* nullptr;

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

        cout *<<* "Enter element " *<<* (i *+* *1*) *<<* ": ";

        string value;

        cin *>>* value;

        auto newNode *=* make\_shared<Node>(value);

*if* (head *==* nullptr) {

            head *=* newNode;

        } *else* {

            tail*->*next *=* newNode;

            newNode*->*prev *=* tail;

        }

        tail *=* newNode;

    }

*return* head;

}

shared\_ptr<Node> addAfter(shared\_ptr<Node> *head*, const string& *targetKey*, const string& *newKey*) {

    auto temp *=* *head*;

*while* (temp *!=* nullptr *&&* temp*->*key *!=* *targetKey*) {

        temp *=* temp*->*next;

    }

*if* (temp *!=* nullptr) {

        auto newNode *=* make\_shared<Node>(*newKey*);

        newNode*->*next *=* temp*->*next;

        newNode*->*prev *=* temp;

*if* (temp*->*next *!=* nullptr) {

            temp*->*next*->*prev *=* newNode;

        }

        temp*->*next *=* newNode;

    }

*return* *head*;

}

shared\_ptr<Node> removeDuplicates(shared\_ptr<Node> *head*) {

    unordered\_set*<*string*>* seen;

    auto current *=* *head*;

    shared\_ptr*<*Node*>* prev *=* nullptr;

*while* (current *!=* nullptr) {

*if* (seen.*find*(current*->*key) *!=* seen.*end*()) {

            auto next *=* current*->*next;

*if* (current*->*next *!=* nullptr) {

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

            }

*if* (prev *!=* nullptr) {

                prev*->*next *=* next;

            } *else* {

*head* *=* next;

            }

            current *=* next;

        } *else* {

            seen.*insert*(current*->*key);

            prev *=* current;

            current *=* current*->*next;

        }

    }

*return* *head*;

}

void printList(shared\_ptr<Node> *head*) {

    auto temp *=* *head*;

*while* (temp *!=* nullptr) {

        cout *<<* temp*->*key *<<* " ";

        temp *=* temp*->*next;

    }

    cout *<<* endl;

}

int main() {

*// Create a doubly linked list*

    cout *<<* "Creating the list:\n";

    auto head *=* createList();

    cout *<<* "Original list: ";

    printList(head);

    head *=* removeDuplicates(head);

    cout *<<* "List after removing duplicates: ";

    printList(head);

    string targetKey, newKey;

    cout *<<* "Enter the target key: ";

    cin *>>* targetKey;

    cout *<<* "Enter the new key to add after the target key: ";

    cin *>>* newKey;

    head *=* addAfter(head, targetKey, newKey);

    cout *<<* "List after adding '" *<<* newKey *<<* "' after '" *<<* targetKey *<<* "': ";

    printList(head);

*return* *0*;

}

**Завдання 2:** Algotester Lab 5 task 2 <https://github.com/artificial-intelligence-department/ai_programming_playground/pull/905/files#diff-345a0aa1e939e21ca191a15148921a07bf814f75b25515ff754c047d1aa183f8>

*#include* <iostream>

*#include* <vector>

*#include* <algorithm>

int main() {

    int N, M;

    std::cin *>>* N *>>* M;

    std::vector*<*std::string*>* cave(N);

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

        std::cin *>>* cave*[*i*]*;

    }

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

        int lastRock *=* N;

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

*if* (cave*[*i*][*j*]* *==* 'X') {

                lastRock *=* i;

            } *else* *if* (cave*[*i*][*j*]* *==* 'S') {

                int k *=* i;

*while* (k*+1* *<* lastRock *&&* cave*[*k*+1][*j*]* *==* 'O') {

                    std::swap(cave*[*k*][*j*]*, cave*[*k*+1][*j*]*);

*++*k;

                }

            }

        }

    }

*for* (const auto*&* row : cave) {

        std::cout *<<* row *<<* std::endl;

    }

*return* *0*;

}

**Завдання 3:** Algotester Lab 7\_8 task 2 <https://github.com/artificial-intelligence-department/ai_programming_playground/pull/905/files#diff-6aa61a0f509250329b3bf8df0220c7db31d751dfec72ef6603483256b381ff61>

*#include* <iostream>

*#include* <sstream>

*#include* <algorithm>

*using* *namespace* std;

template <class T>

class DynamicArray {

private:

    T*\** data;

    size\_t size;

    size\_t capacity;

    const int growthFactor *=* *2*;

    void *resize*() {

*do* {

            capacity *\*=* growthFactor;

        } *while* (capacity *<=* size);

        T*\** newData *=* *new* T[capacity];

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

            newData[i] *=* data[i];

        }

*delete[]* data;

        data *=* newData;

    }

public:

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

        data *=* *new* T[capacity];

    }

*~DynamicArray*() {

*delete[]* data;

    }

    void *insert*(int *index*, size\_t *N*, T\* *values*) {

*if* (*index* *>* size) {

*return*;

        }

        size *+=* *N*;

*while* (size *>=* capacity) {

*resize*();

        }

        copy\_backward(data *+* *index*, data *+* size *-* *N*, data *+* size);

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

            data[*index* *+* i] *=* *values*[i];

        }

    }

    void *erase*(int *index*, size\_t *n*) {

*if* (*index* *>=* size) {

*return*;

        }

        size\_t elementsToRemove *=* min(*n*, size *-* *index*);

*for* (size\_t i *=* *index*; i *<* size *-* elementsToRemove; *++*i) {

            data[i] *=* data[i *+* elementsToRemove];

        }

        size *-=* elementsToRemove;

    }

    size\_t *getSize*() const {

*return* size;

    }

    size\_t *getCapacity*() const {

*return* capacity;

    }

    T& *operator[]*(int *index*) {

*if* (*index* *<* *0* *||* *index* *>=* size) {

*throw* out\_of\_range("Index out of range");

        }

*return* data[*index*];

    }

    void *set*(int *index*, T *value*) {

*if* (*index* *<* *0* *||* *index* *>=* size) {

*throw* out\_of\_range("Index out of range");

        }

        data[*index*] *=* *value*;

    }

    void *print*() const {

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

            cout *<<* data[i] *<<* " ";

        }

        cout *<<* endl;

    }

};

int main() {

    int Q;

    cin *>>* Q;

    DynamicArray*<*int*>* dynamicArray;

    string*\** commands *=* *new* string[Q];

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

        cin *>>* commands[i];

*if* (commands[i] *==* "size") {

            cout *<<* dynamicArray.*getSize*() *<<* endl;

        }

*else* *if* (commands[i] *==* "capacity") {

            cout *<<* dynamicArray.*getCapacity*() *<<* endl;

        }

*else* *if* (commands[i] *==* "insert") {

            int index, N;

            cin *>>* index *>>* N;

            int*\** values *=* *new* int[N];

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

                cin *>>* values[j];

            }

            dynamicArray.*insert*(index, N, values);

*delete[]* values;

        }

*else* *if* (commands[i] *==* "erase") {

            int index, n;

            cin *>>* index *>>* n;

            dynamicArray.*erase*(index, n);

        }

*else* *if* (commands[i] *==* "set") {

            int index, value;

            cin *>>* index *>>* value;

            dynamicArray.*set*(index, value);

        }

*else* *if* (commands[i] *==* "get") {

            int index;

            cin *>>* index;

            cout *<<* dynamicArray*[*index*]* *<<* endl;

        }

*else* *if* (commands[i] *==* "print") {

            dynamicArray.*print*();

        }

    }

*delete[]* commands;

*return* *0*;

}

**Завдання 4:** Class Practice Task <https://github.com/artificial-intelligence-department/ai_programming_playground/pull/905/files#diff-728da670b74a9258b10e017a13d177f8140f247997393bf3a479eefb7cbe73c9>

*#include* <iostream>

*#include* <memory>

*using* *namespace* std;

struct Node {

    int data;

    shared\_ptr*<*Node*>* next;

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

};

shared\_ptr<Node> createList() {

    cout *<<* "Enter the number of elements in the list: ";

    int n;

    cin *>>* n;

    shared\_ptr*<*Node*>* head *=* nullptr;

    shared\_ptr*<*Node*>* tail *=* nullptr;

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

        cout *<<* "Enter element " *<<* (i *+* *1*) *<<* ": ";

        int value;

        cin *>>* value;

        auto newNode *=* make\_shared<Node>(value);

*if* (head *==* nullptr) {

            head *=* newNode;

        } *else* {

            tail->next *=* newNode;

        }

        tail *=* newNode;

    }

*return* head;

}

bool compare(const shared\_ptr<Node>& h1, const shared\_ptr<Node>& h2) {

    auto n1 *=* h1;

    auto n2 *=* h2;

*while* (n1 *!=* nullptr *&&* n2 *!=* nullptr) {

*if* (n1->data *!=* n2->data) {

*return* false; *// Data mismatch detected*

        }

        n1 *=* n1->next;

        n2 *=* n2->next;

    }

*// Check if both lists ended at the same time (lists are of equal length)*

*return* n1 *==* nullptr *&&* n2 *==* nullptr;

}

int main() {

*// Example usage*

    cout *<<* "Creating the first list:\n";

    auto head1 *=* createList();

    cout *<<* "Creating the second list:\n";

    auto head2 *=* createList();

*if* (compare(head1, head2)) {

        cout *<<* "The lists are identical." *<<* endl;

    } *else* {

        cout *<<* "The lists are different." *<<* endl;

    }

*return* *0*;

}

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

VNS Lab 10

Зображення, що містить текст, знімок екрана, Шрифт, дизайн

Автоматично згенерований опис

Рисунок

Algotester Lab 5 task 2

Зображення, що містить текст, знімок екрана, Шрифт

Автоматично згенерований опис

Рисунок

Зображення, що містить текст, Шрифт, ряд, число

Автоматично згенерований опис

Рисунок

Algotester Lab 7\_8 task 2

Зображення, що містить знімок екрана, текст, Шрифт, дизайн

Автоматично згенерований опис

Рисунок

Зображення, що містить текст, знімок екрана, Шрифт, число

Автоматично згенерований опис

Рисунок

Class Practice Task

Зображення, що містить текст, знімок екрана, Шрифт

Автоматично згенерований опис

Рисунок

# **Висновок:**

Детально вивчив створення однозв'язних та двозв'язних списків. Нарешті усвідомив, для чого потрібні поінтери і чому не варто виводити їх через cout (пробував вивести linked list через cout). Потребувало багато часу для розуміння логіки з'єднання списків (зміни голови, хвоста, head->next, head->prev). Зустрівся з референсом на поінтер (Node\*&) вперше, виглядало страшно, але виявилося корисним, особливо при змінах в самому списку.

Також навчився впораватися зі стресом, коли не міг розв'язати завдання протягом довгого часу. Особливо це стосувалося завдання з алготестеру, де потрібно було організувати чергу піску. Навіть якщо це була відносно легка задача, неспроможність вирішити її призвела до стресу, що вплинуло на продуктивність. Це вчило мене управляти своєю емоційною реакцією і робити раціональні рішення, не дозволяючи стресу впливати на ефективність роботи.

Глибше освоїв структури даних, зрозумів їх призначення і запам'ятав синтаксис. Навіть створив свій клас, хоча він був простий, це все ж дало мені можливість практично застосувати теоретичні знання.

І найголовніше - розвинув навички управління часом і усвідомив, що краще не відкладати все до останнього моменту.