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

**Виконала:**

Студентка групи ШІ-14

Михальчук Віра Іванівна

Львів 2023

# **Тема роботи:**

Ознайомлення з динамічними структурами та їхніми алгоритмами обробки. Практичне застосування знань на основі виконання лабораторних та практичних робіт.

# **Мета роботи:**

Ознайомитися з вище переліченими списками тем, удосконалити свої знання в теорії та практичному застосуванні вивченого.

# **Теоретичні відомості:**

1. Теоретичні відомості з переліком важливих тем:

* **Тема №1:** Динамічний масив.
* **Тема №2:** Бінарне дерево.
* **Тема №3:** Двозв’язний та однозв’язний списки.

1. Індивідуальний план опрацювання теорії:

* Тема **№1: Динамічний масив.**
  + Джерела Інформації:
    - Відео.

Відео-теорія на теми:

[Динамічний масив](https://youtu.be/fk4YTLsnKmw?t=1097)

[Динамічний масив](https://youtu.be/oEppO5GH5ic?t=640)

* + - Статті.

[Динамічні масиви.](https://acode.com.ua/urok-90-dynamichni-masyvy/)

[Динамічний](http://cpp.dp.ua/dynamichni-masyvy/) масив.

* + Що опрацьовано:

1. Динамічний масив.
2. Ініціалізація динамічного масиву
3. Видалення динамічного масиву.
   * Статус: Ознайомлена
   * Початок опрацювання теми: 20.12.2023
   * Звершення опрацювання теми: 20.12.2023

* Тема **№2: Бінарне дерево.**
  + Джерела Інформації:
    - Відео.

Відео-теорія про:

[Бінарне дерево.](https://youtu.be/qBFzNW0ALxQ?t=4563)

[Бінарне дерево](https://youtu.be/9o_i0zzxk1s?t=959)

* + - Стаття.

[Бінарне дерево.](https://uk.myservername.com/binary-search-tree-c)

* + Що опрацьовано:

1. Що таке бінарне дерево.
2. Реалізація бінарного дерева.
   * Статус: Ознайомлена
   * Початок опрацювання теми: 20.12.2023
   * Звершення опрацювання теми: 20.12.2023

* Тема **№3: Двозв’язний та однозв’язний списки.**
  + Джерела Інформації:
    - Відео.

[Однозв’язний список.](https://youtu.be/-25REjF_atI?t=5660)

[Двозв’язний список.](https://youtu.be/QLzu2-_QFoE?t=5385) (Блоган, і ще одна весела нічка )

Що опрацьовано:

1. Однозв’язний список, його пряме використання.
2. Двозв’язний список, його пряме використання
   * Статус: Ознайомлена
   * Початок опрацювання теми: 20.12.2023
   * Звершення опрацювання теми: 16.12.2023

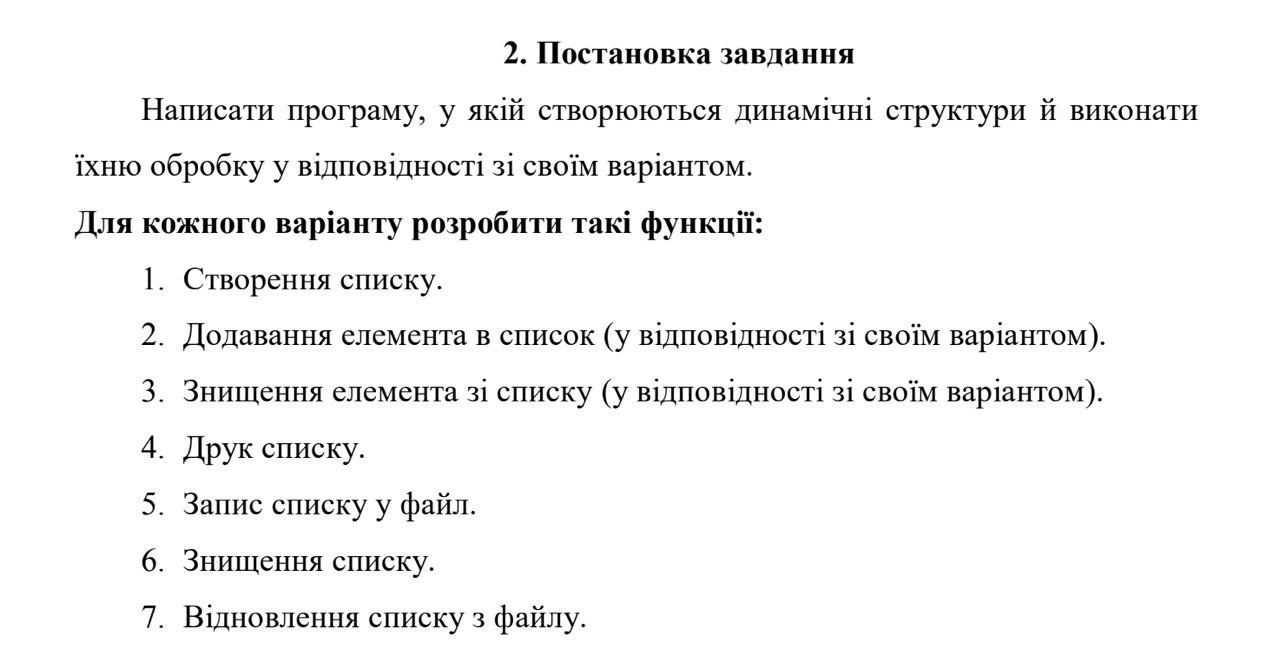
* Тема **№4 Черга**
  + Джерела Інформації:
    - <https://www.youtube.com/watch?v=jaK4pn1jXTo&list=PL43pGnjiVwgS5njI0HMGnqSH18tSSuLz_&index=6&pp=iAQB>
  + Що опрацьовано:
    - Ознайомилась з теорією
  + Статус: Ознайомлена
  + Початок опрацювання теми: 20.12.23
  + Завершення опрацювання теми: 20.12.23

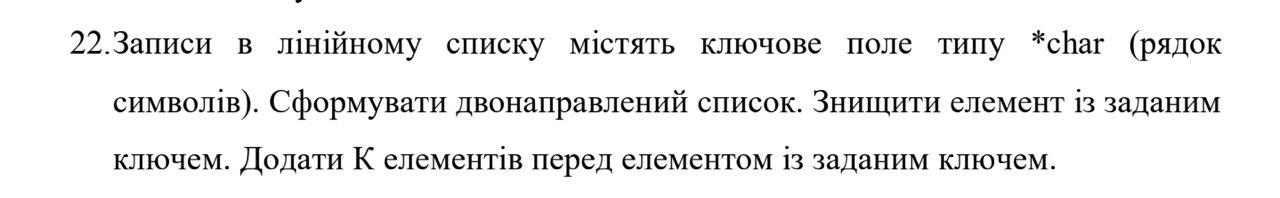
# **Виконання роботи:**

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

Завдання №1 VNS Lab 10

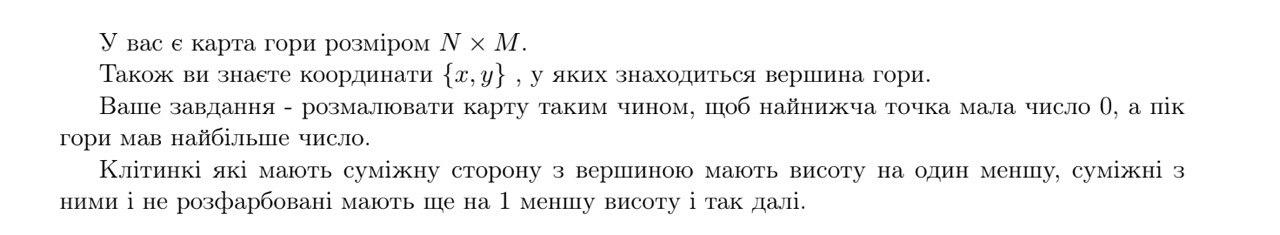
* Варіант 22
* Деталі завдання





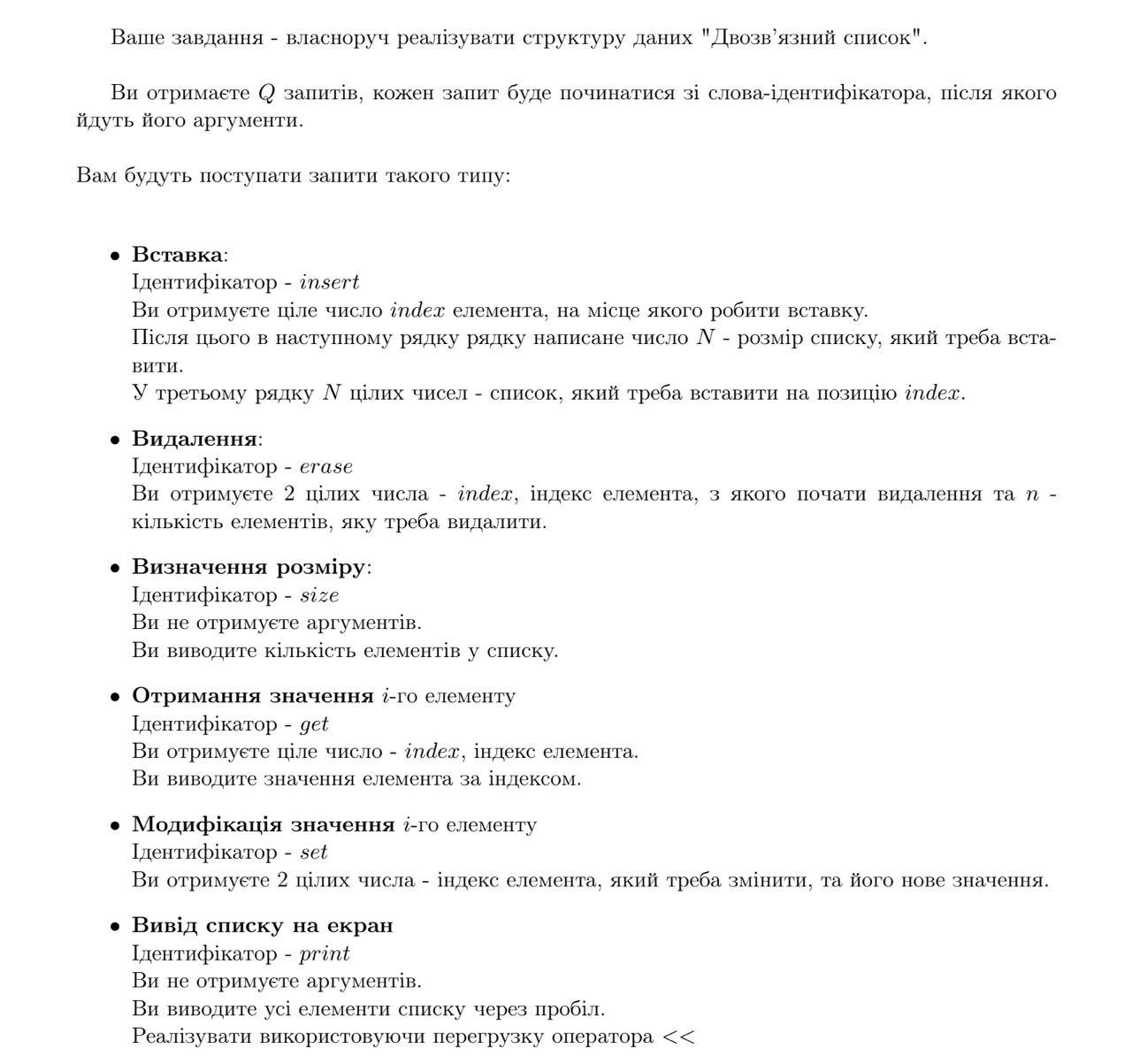
Завдання №2 Algotester Lab 5

* Варіант 3
* Деталі завдання



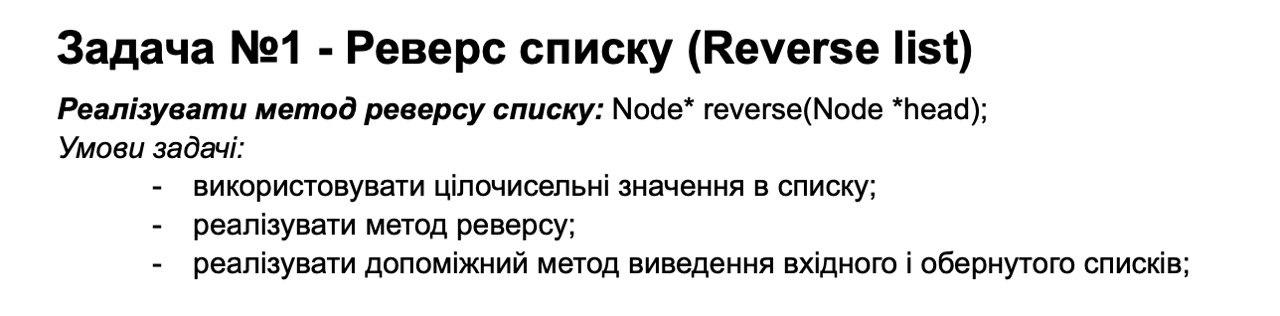
Завдання №3 Algotester Lab7-8

* Варіант 1
* Деталі завдання



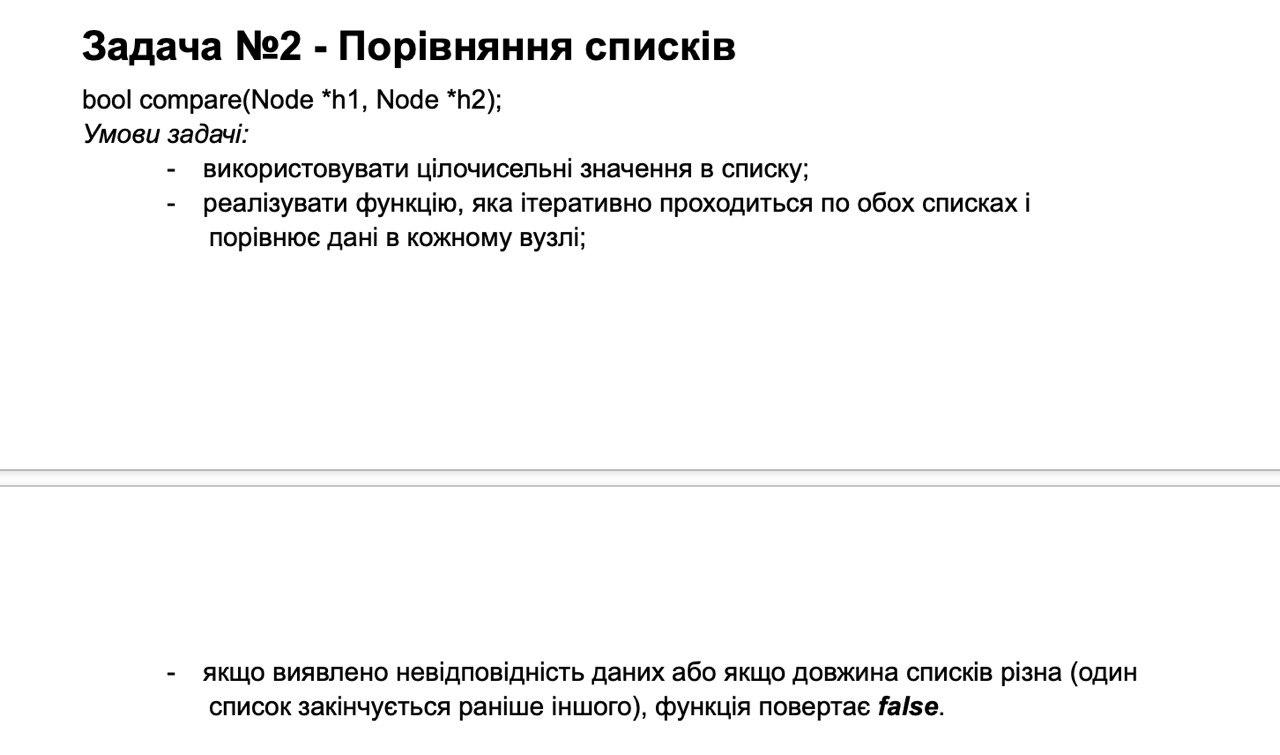
Завдання №4 Class Practice Work Task 1

* Деталі завдання



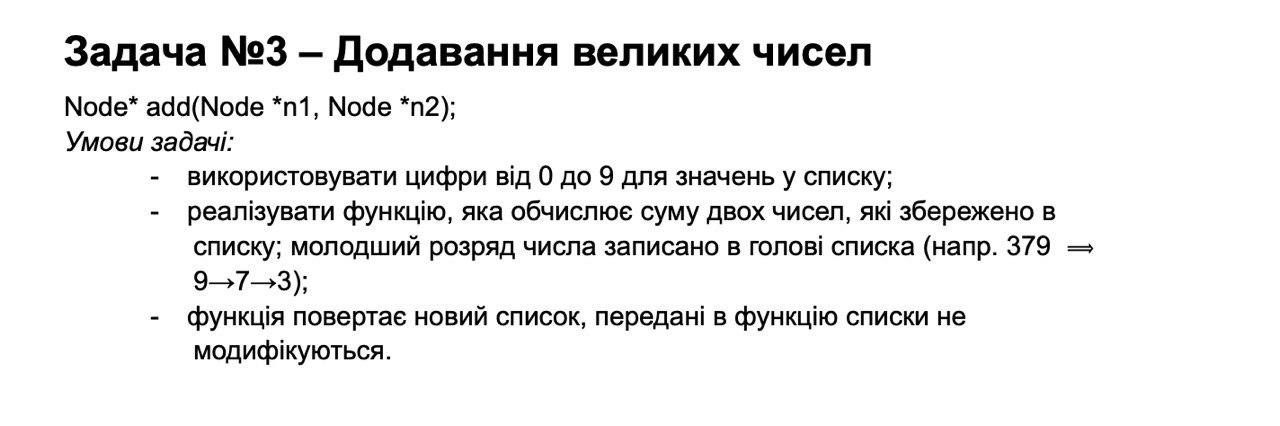
Завдання №5 Class Practice Work Task 2

* Деталі завдання



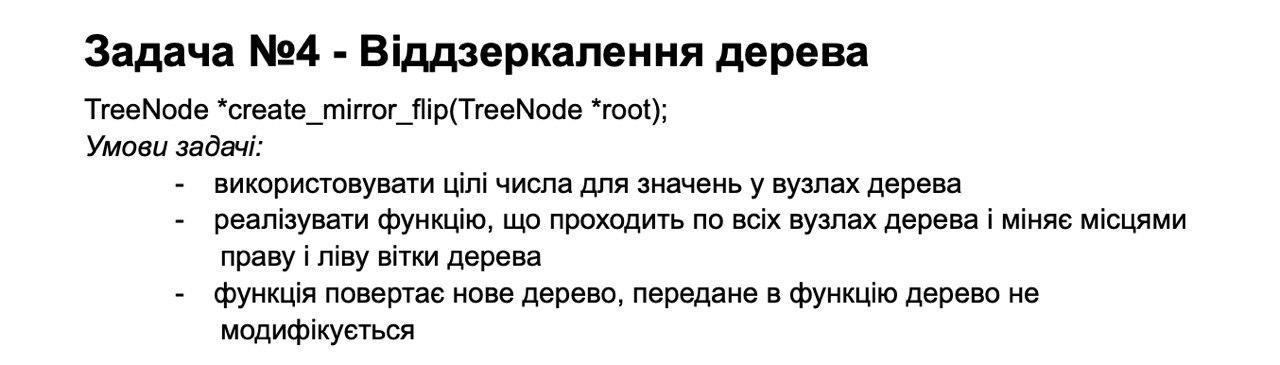
Завдання №6 Class Practice Work Task 3

* Деталі завдання



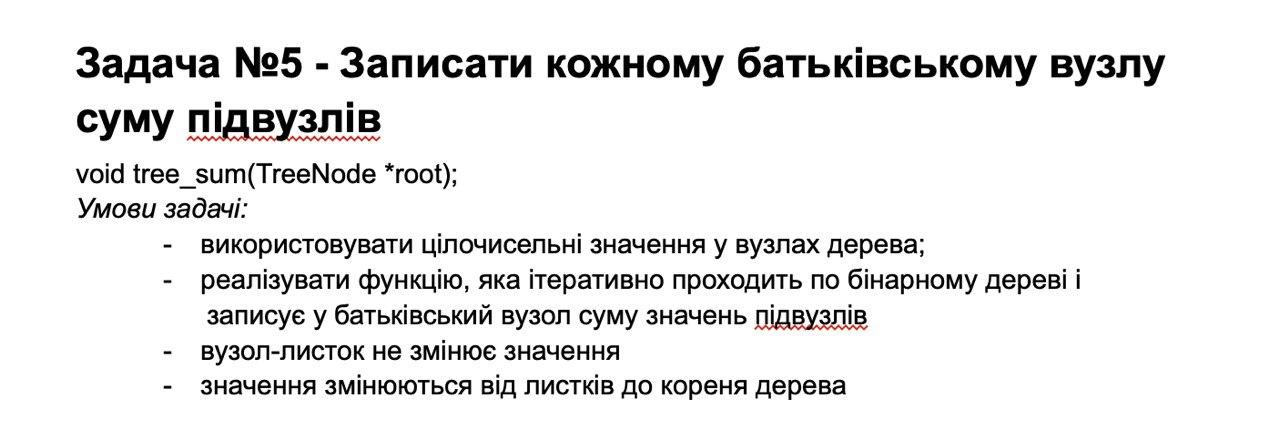
Завдання №7 Class Practice Work Task 4

* Деталі завдання



Завдання №8 Class Practice Work Task 5

* Деталі завдання



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

Програма №1 VNS Lab 10

* Планований час на реалізацію 1 година

Програма №2 Algotester Lab 5

* Планований час на реалізацію 1.5 години

Програма №3 Algotester Lab7-8

* Планований час на реалізацію 1.5 години

Програма №4 Class Practice Work Task 1

* Планований час на реалізацію 40 хв

Програма №5 Class Practice Work Task 2

* Планований час на реалізацію 40 хв

Програма №6 Class Practice Work Task 3

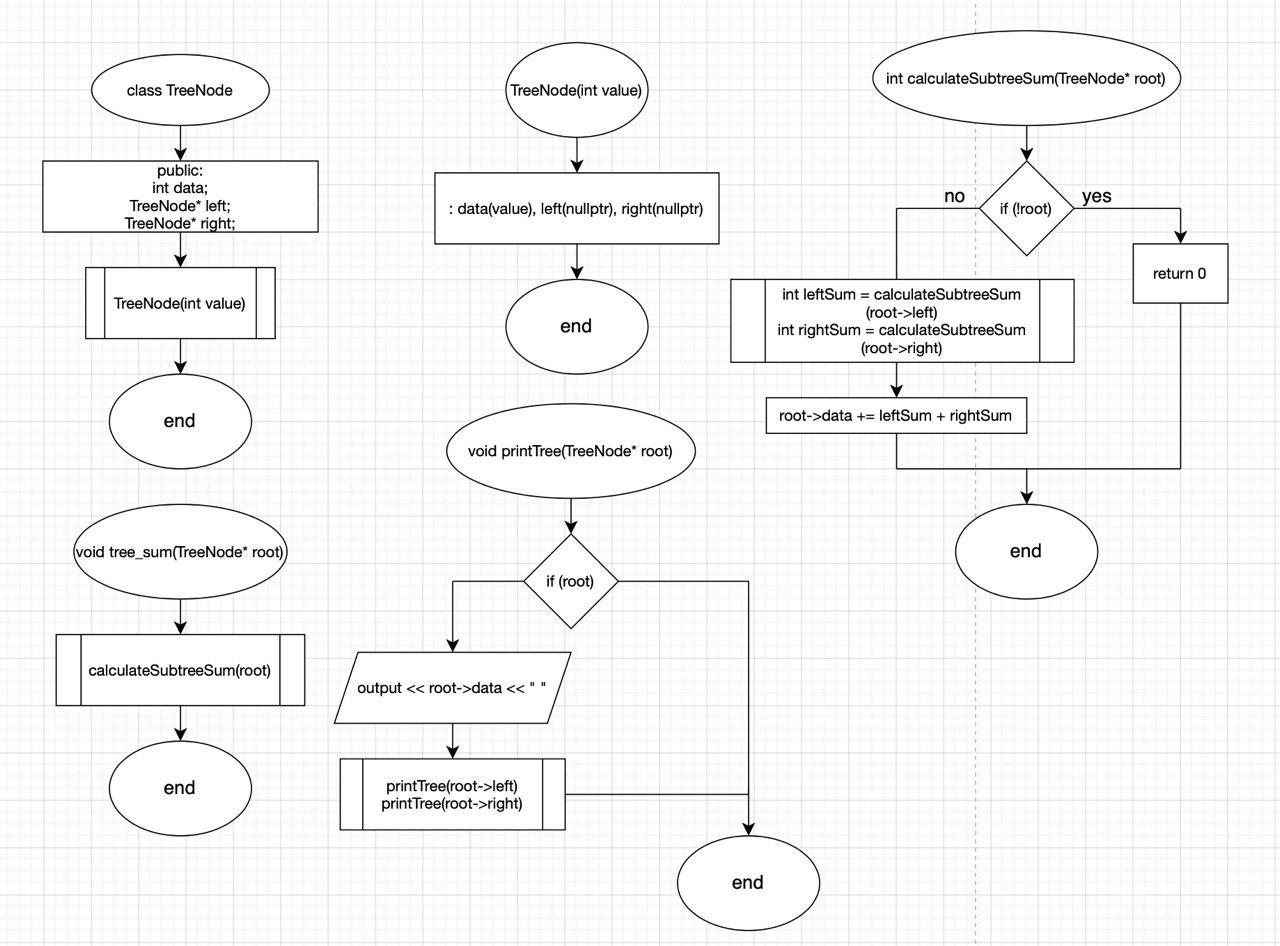
* Планований час на реалізацію 40 хв

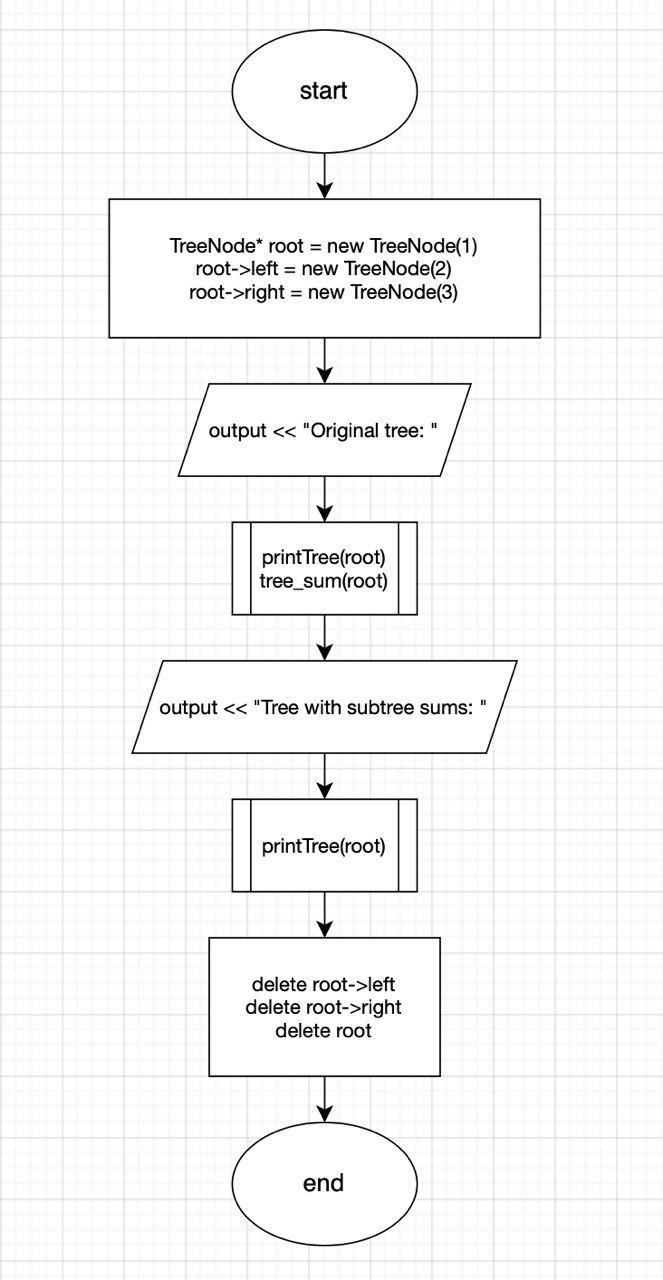
Програма №7 Class Practice Work Task 4

* Планований час на реалізацію 40 хв

Програма №8 Class Practice Work Task 5

* Блок-схема





* Планований час на реалізацію 1 година

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

Завдання №1 VNS Lab 10

[vns\_lab\_10\_vira\_mykhalchuk.cpp](https://github.com/artificial-intelligence-department/ai_programming_playground/blob/epic_6_practice_and_labs_vira_mykhalchuk/ai_14/vira_mykhalchuk/epic_6_practice_and_labs_vira_mykhalchuk/vns_lab_10_vira_mykhalchuk.cpp)

#include <iostream>

#include <fstream>

#include <cstring>

#include <sstream>

#include <string>

using namespace std;

struct Node

{

char\* data;

Node\* previous;

Node\* next;

};

Node\* createNewList()

{

return nullptr;

}

void add(Node\* &head, const char\* element)

{

Node\* newNode = new Node;

newNode->data = new char[strlen(element) + 1];

strcpy(newNode->data, element);

newNode->next = nullptr;

if (head == nullptr)

{

head = newNode;

newNode->previous = nullptr;

}

else

{

Node\* temporary = head;

while (temporary->next != nullptr)

{

temporary = temporary->next;

}

temporary->next = newNode;

newNode->previous = temporary;

}

}

void print(Node\* head)

{

if (head == nullptr)

{

cout << "The list is empty" << endl;

return;

}

Node\* current = head;

while (current != nullptr)

{

cout << current->data << " ";

current = current->next;

}

cout << endl;

}

void deleteElement(Node\* &head, const char\* key)

{

Node\* current = head;

while (current != nullptr && strcmp(current->data, key) != 0)

{

current = current->next;

}

if (current == nullptr)

{

cout << "Element with the key " << key << " isn't found!" << endl;

return;

}

if (current->previous != nullptr)

{

current->previous->next = current->next;

}

else

{

head = current->next;

}

if (current->next != nullptr)

{

current->next->previous = current->previous;

}

delete[] current->data;

delete current;

}

void addKElemBeforeKey(Node\* &head, const char\* key, int k, const char\* elements)

{

Node\* current = head;

while (current != nullptr && strcmp(current->data, key) != 0)

{

current = current->next;

}

if (current == nullptr)

{

cout << "Element with the key " << key << " isn't found!" << endl;

return;

}

istringstream iss(elements);

string element;

for (int i = 0; i < k && iss >> element; ++i)

{

Node\* newNode = new Node;

newNode->data = new char[element.length() + 1];

strcpy(newNode->data, element.c\_str());

if (current->previous != nullptr)

{

current->previous->next = newNode;

}

else

{

head = newNode;

}

newNode->previous = current->previous;

newNode->next = current;

if (current->previous != nullptr)

{

current->previous = newNode;

}

}

}

void writeToFile(Node\* head, const char\* nameOfFile)

{

ofstream file\_output(nameOfFile);

if (!file\_output.is\_open())

{

cout << "File couldn't be opened!" << endl;

return;

}

Node\* current = head;

while (current != nullptr)

{

file\_output << current->data << endl;

current = current->next;

}

cout << "List is written to file '" << nameOfFile << "'" << endl;

file\_output.close();

}

void deleteList(Node\* &head)

{

while (head != nullptr)

{

Node\* temporary = head;

head = head->next;

delete[] temporary->data;

delete temporary;

}

}

Node\* restoreList(const char\* nameOfFile)

{

ifstream file\_input(nameOfFile);

Node\* head = nullptr;

if (!file\_input.is\_open())

{

cout << "File couldn't be opened!" << endl;

return nullptr;

}

string line;

while (getline(file\_input, line))

{

add(head, line.c\_str());

}

file\_input.close();

cout << "List is restored from file '" << nameOfFile << "'" << endl;

return head;

}

int main()

{

Node\* List = createNewList();

string action;

do

{

cout << endl;

cout << "Choose the operation (enter an action): " << endl;

cout << "1. To add element write: \"add element\"" << endl;

cout << "2. To delete element write: \"delete element\"" << endl;

cout << "3. To add element before another element write: \"add before\"" << endl;

cout << "4. To print the whole list write: \"print\"" << endl;

cout << "5. To write list to file write: \"write\"" << endl;

cout << "6. To restore list from file write: \"restore list\"" << endl;

cout << "7. T delete the entire list write: \"delete list\"" << endl;

cout << "0. To exit write: \"exit\"" << endl;

cout << "Enter the action -> ";

getline(cin, action);

if (action == "add element")

{

cout << "Enter the element you want to add: ";

string element;

getline(cin, element);

add(List, element.c\_str());

}

else if (action == "delete element")

{

cout << "Enter the key of the element you want to delete: ";

string key;

getline(cin, key);

deleteElement(List, key.c\_str());

}

else if (action == "add before")

{

cout << "Enter the key before which you want to add elements: ";

string key;

getline(cin, key);

int k;

cout << "Enter the number of elements you want to add: ";

cin >> k;

cin.ignore();

cout << "Enter the elements you want to add: ";

string ELEMENTS;

getline(cin, ELEMENTS);

addKElemBeforeKey(List, key.c\_str(), k, ELEMENTS.c\_str());

}

else if (action == "print")

{

cout << "Your current list: ";

print(List);

}

else if (action == "write")

{

cout << "Enter the name of the file in which you want to write the list: ";

string fileName;

getline(cin, fileName);

writeToFile(List, fileName.c\_str());

}

else if (action == "restore list")

{

cout << "Enter the name of the file from which you want to restore the list: ";

string fileName;

getline(cin, fileName);

deleteList(List);

List = restoreList(fileName.c\_str());

print(List);

}

else if (action == "delete list")

{

deleteList(List);

List = createNewList();

cout << "The entire list now is deleted!" << endl;

}

else if (action != "exit")

{

cout << "Action not found! Try again." << endl;

}

} while (action != "exit");

deleteList(List);

return 0;

}

Завдання №2 Algotester Lab 5

[algotester\_lab\_5\_v\_3\_vira\_mykhalchuk.cpp](https://github.com/artificial-intelligence-department/ai_programming_playground/blob/epic_6_practice_and_labs_vira_mykhalchuk/ai_14/vira_mykhalchuk/epic_6_practice_and_labs_vira_mykhalchuk/algotester_lab_5_v_3_vira_mykhalchuk.cpp)

#include <iostream>

#include <cmath>

using namespace std;

int main() {

int N, M, x, y;

cin >> N >> M >> x >> y;

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

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

heights[i] = new int[M]();

}

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

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

heights[i][j] = -abs(x - 1 - i) - abs(y - 1 - j);

}

}

int min\_height = 0;

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

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

if (min\_height > heights[i][j]) {

min\_height = heights[i][j];

}

}

}

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

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

heights[i][j] -= min\_height;

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

}

cout << endl;

}

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

delete[] heights[i];

}

delete[] heights;

return 0;

}

Завдання №3 Algotester Lab7-8

[algotester\_lab\_78\_v\_1\_vira\_mykhalchuk.cpp](https://github.com/artificial-intelligence-department/ai_programming_playground/blob/epic_6_practice_and_labs_vira_mykhalchuk/ai_14/vira_mykhalchuk/epic_6_practice_and_labs_vira_mykhalchuk/algotester_lab_78_v_1_vira_mykhalchuk.cpp)

#include <iostream>

#include <string>

class LinkedList {

private:

struct Node {

int value;

Node\* prev;

Node\* next;

Node(int val, Node\* previous = nullptr, Node\* nextNode = nullptr)

: value(val), prev(previous), next(nextNode) {}

};

Node\* head;

Node\* tail;

public:

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

~LinkedList() {

clear();

}

void insert(int index, int N, int arr[]) {

if (index > size()) {

return;

}

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

Node\* newNode = new Node(arr[i]);

if (index == 0) {

newNode->next = head;

if (head) {

head->prev = newNode;

} else {

tail = newNode;

}

head = newNode;

} else if (index == size()) {

newNode->prev = tail;

if (tail) {

tail->next = newNode;

} else {

head = newNode;

}

tail = newNode;

} else {

Node\* current = getNode(index - 1);

newNode->next = current->next;

newNode->prev = current;

if (current->next) {

current->next->prev = newNode;

}

current->next = newNode;

}

}

}

void erase(int index, int N) {

if (index < 0 || N <= 0) {

return;

}

Node\* current = getNode(index);

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

Node\* nextNode = current->next;

Node\* prevNode = current->prev;

if (prevNode) {

prevNode->next = nextNode;

} else {

head = nextNode;

}

if (nextNode) {

nextNode->prev = prevNode;

} else {

tail = prevNode;

}

delete current;

current = nextNode;

}

}

int size() const {

int size = 0;

Node\* current = head;

while (current) {

++size;

current = current->next;

}

return size;

}

void set(int index, int content) {

Node\* current = getNode(index);

if (current) {

current->value = content;

}

}

int get(int index) const {

Node\* current = getNode(index);

return current ? current->value : 0;

}

void print() const {

Node\* current = head;

while (current) {

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

current = current->next;

}

std::cout << std::endl;

}

private:

Node\* getNode(int index) const {

Node\* current = head;

for (int i = 0; i < index && current; ++i) {

current = current->next;

}

return current;

}

void clear() {

while (head) {

Node\* temp = head;

head = head->next;

delete temp;

}

tail = nullptr;

}

};

int main() {

int Q;

std::cin >> Q;

LinkedList list;

while (Q--) {

std::string request;

std::cin >> request;

if (request == "insert") {

int index, N;

std::cin >> index >> N;

int nums[N];

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

std::cin >> nums[i];

}

list.insert(index, N, nums);

} else if (request == "erase") {

int index, n;

std::cin >> index >> n;

list.erase(index, n);

} else if (request == "size") {

std::cout << list.size() << std::endl;

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

int index, content;

std::cin >> index >> content;

list.set(index, content);

} else if (request == "get") {

int index;

std::cin >> index;

std::cout << list.get(index) << std::endl;

} else if (request == "print") {

list.print();

}

}

return 0;

}

Завдання №4 Class Practice Work Task 1

[class\_practice\_task\_1\_vira\_mykhalchuk.cpp](https://github.com/artificial-intelligence-department/ai_programming_playground/blob/epic_6_practice_and_labs_vira_mykhalchuk/ai_14/vira_mykhalchuk/epic_6_practice_and_labs_vira_mykhalchuk/class_practice_task_1_vira_mykhalchuk.cpp)

#include <iostream>

using namespace std;

class Node {

public:

int data;

Node\* next;

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

};

class LinkedList {

public:

Node\* head;

LinkedList() : head(nullptr) {}

void append(int value) {

Node\* newNode = new Node(value);

if (!head) {

head = newNode;

return;

}

Node\* current = head;

while (current->next) {

current = current->next;

}

current->next = newNode;

}

Node\* reverse(Node\* head) {

Node\* prev = nullptr;

Node\* current = head;

Node\* nextNode = nullptr;

while (current) {

nextNode = current->next;

current->next = prev;

prev = current;

current = nextNode;

}

return prev;

}

void printList(Node\* list) const {

Node\* current = list;

while (current) {

cout << current->data << " ";

current = current->next;

}

cout << endl;

}

};

int main() {

LinkedList myList;

myList.append(1);

myList.append(2);

myList.append(3);

myList.append(4);

myList.append(5);

cout << "Original list: ";

myList.printList(myList.head);

myList.head = myList.reverse(myList.head);

cout << "Reversed list: ";

myList.printList(myList.head);

return 0;

}

Завдання №5 Class Practice Work Task 2

[class\_practice\_task\_2\_vira\_mykhalchuk.cpp](https://github.com/artificial-intelligence-department/ai_programming_playground/blob/epic_6_practice_and_labs_vira_mykhalchuk/ai_14/vira_mykhalchuk/epic_6_practice_and_labs_vira_mykhalchuk/class_practice_task_2_vira_mykhalchuk.cpp)

#include <iostream>

using namespace std;

class Node {

public:

int data;

Node\* next;

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

};

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

while (h1 && h2) {

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);

if (compare(head1, head2)) {

cout << "Lists are the same." << endl;

} else {

cout << "Lists are different." << endl;

}

while (head1) {

Node\* temp = head1;

head1 = head1->next;

delete temp;

}

while (head2) {

Node\* temp = head2;

head2 = head2->next;

delete temp;

}

return 0;

}

Завдання №6 Class Practice Work Task 3

[class\_practice\_task\_3\_vira\_mykhalchuk.cpp](https://github.com/artificial-intelligence-department/ai_programming_playground/blob/epic_6_practice_and_labs_vira_mykhalchuk/ai_14/vira_mykhalchuk/epic_6_practice_and_labs_vira_mykhalchuk/class_practice_task_3_vira_mykhalchuk.cpp)

#include <iostream>

using namespace std;

class Node {

public:

int data;

Node\* next;

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

};

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

Node\* resultHead = nullptr;

Node\* currentResult = nullptr;

int carry = 0;

while (n1 || n2 || carry) {

int sum = (n1 ? n1->data : 0) + (n2 ? n2->data : 0) + carry;

carry = sum / 10;

sum %= 10;

Node\* newNode = new Node(sum);

if (!resultHead) {

resultHead = newNode;

currentResult = resultHead;

} else {

currentResult->next = newNode;

currentResult = newNode;

}

if (n1) n1 = n1->next;

if (n2) n2 = n2->next;

}

return resultHead;

}

void printList(Node\* list) {

while (list) {

cout << list->data;

list = list->next;

if (list) cout << " -> ";

}

cout << endl;

}

int main() {

Node\* number1 = new Node(4);

number1->next = new Node(3);

number1->next->next = new Node(2);

number1->next->next->next = new Node(1);

Node\* number2 = new Node(8);

number2->next = new Node(7);

number2->next->next = new Node(6);

number2->next->next->next = new Node(5);

Node\* result = add(number1, number2);

cout << "Число 1: ";

printList(number1);

cout << "Число 2: ";

printList(number2);

cout << "Сума: ";

printList(result);

while (number1) {

Node\* temp = number1;

number1 = number1->next;

delete temp;

}

while (number2) {

Node\* temp = number2;

number2 = number2->next;

delete temp;

}

while (result) {

Node\* temp = result;

result = result->next;

delete temp;

}

return 0;

}

Завдання №7 Class Practice Work Task 4

[class\_practice\_task\_4\_vira\_mykhalchuk.cpp](https://github.com/artificial-intelligence-department/ai_programming_playground/blob/epic_6_practice_and_labs_vira_mykhalchuk/ai_14/vira_mykhalchuk/epic_6_practice_and_labs_vira_mykhalchuk/class_practice_task_4_vira_mykhalchuk.cpp)

#include <iostream>

using namespace std;

class TreeNode {

public:

int data;

TreeNode\* left;

TreeNode\* right;

TreeNode(int value) : data(value), left(nullptr), right(nullptr) {}

};

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

if (!root) {

return nullptr;

}

TreeNode\* newRoot = new TreeNode(root->data);

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

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

return newRoot;

}

void destroyTree(TreeNode\* root) {

if (root) {

destroyTree(root->left);

destroyTree(root->right);

delete root;

}

}

void printTree(TreeNode\* root) {

if (root) {

cout << root->data << " ";

printTree(root->left);

printTree(root->right);

}

}

int main() {

TreeNode\* root = new TreeNode(0);

root->left = new TreeNode(50);

root->right = new TreeNode(100);

root->left->left = new TreeNode(3);

root->left->right = new TreeNode(80);

cout << "Original tree: ";

printTree(root);

cout << endl;

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

cout << "Mirrored tree: ";

printTree(mirroredRoot);

cout << endl;

destroyTree(root);

destroyTree(mirroredRoot);

return 0;

}

Завдання №8 Class Practice Work Task 5

[class\_practice\_task\_5\_vira\_mykhalchuk.cpp](https://github.com/artificial-intelligence-department/ai_programming_playground/blob/epic_6_practice_and_labs_vira_mykhalchuk/ai_14/vira_mykhalchuk/epic_6_practice_and_labs_vira_mykhalchuk/class_practice_task_5_vira_mykhalchuk.cpp)

#include <iostream>

using namespace std;

class TreeNode {

public:

int data;

TreeNode\* left;

TreeNode\* right;

TreeNode(int value) : data(value), left(nullptr), right(nullptr) {}

};

int calculateSubtreeSum(TreeNode\* root) {

if (!root) {

return 0;

}

int leftSum = calculateSubtreeSum(root->left);

int rightSum = calculateSubtreeSum(root->right);

root->data += leftSum + rightSum;

return root->data;

}

void tree\_sum(TreeNode\* root) {

calculateSubtreeSum(root);

}

void printTree(TreeNode\* root) {

if (root) {

cout << root->data << " ";

printTree(root->left);

printTree(root->right);

}

}

int main() {

TreeNode\* root = new TreeNode(1);

root->left = new TreeNode(2);

root->right = new TreeNode(3);

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

cout << "Original tree: ";

printTree(root);

cout << endl;

tree\_sum(root);

cout << "Tree with subtree sums: ";

printTree(root);

cout << endl;

delete root->left;

delete root->right;

delete root;

return 0;

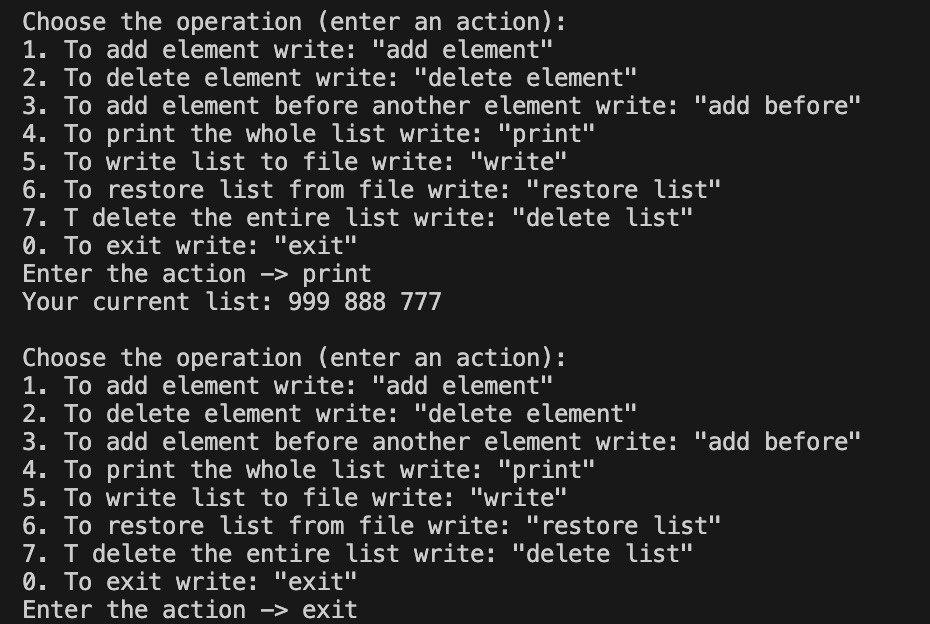
}

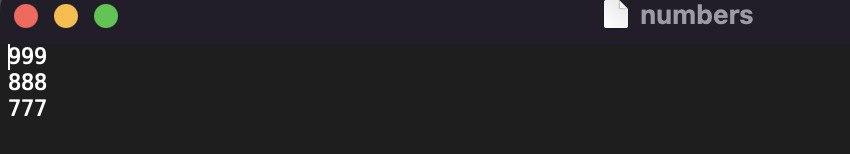
## **4. Результати виконання завдань, тестування та фактично затрачений час:**

Завдання №1 VNS Lab 10









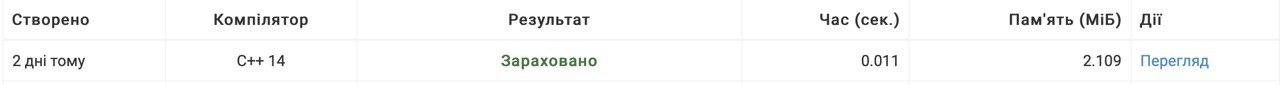
Час затрачений на виконання завдання 1.5 години

Завдання №2 Algotester Lab 5



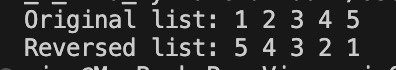
Час затрачений на виконання завдання 1 година

Завдання №3 Algotester Lab7-8



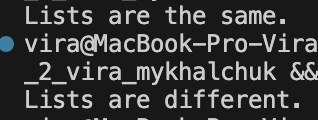
Час затрачений на виконання завдання 1 година

Завдання №4 Class Practice Work Task 1



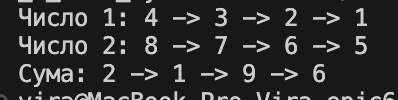
Час затрачений на виконання завдання 40 хв

Завдання №5 Class Practice Work Task 2



Час затрачений на виконання завдання 50 хв

Завдання №6 Class Practice Work Task 3



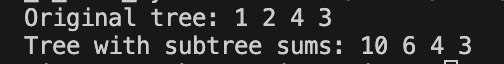
Час затрачений на виконання завдання 1 година

Завдання №7 Class Practice Work Task 4



Час затрачений на виконання завдання 1 година

Завдання №8 Class Practice Work Task 5



Час затрачений на виконання завдання 1 година 20 хв

# **Висновки:**

Вивчення динамічних структур даних та алгоритмів обробки є захоплюючим кроком у розвитку моїх програмістських навичок. Ця тема допомагає мені ефективно працювати з обробкою та управлінням даними, розширюючи мої можливості створення оптимізованих програм. Отримані навички вибору та використання структур даних сприяють реалізації успішних програмних рішень у моїх проектах.