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(ла):**

Студент групи ШІ-11

Попович Олег Васильович

# **Тема роботи:**

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

# **Мета роботи:**

Ознайомитися з динамічними структурами (Черга, Стек, Списки, Дерево). Алгоритми обробки динамічних структур

Виконати задані лабораторні роботи.

Виконати задане практичне завдання.

# **Теоретичні відомості:**

1. Теоретичні відомості з переліком важливих тем:

* Тема №1: Списки
* Тема №2: Стек, Черга
* Тема №3: Дерево

1. Індивідуальний план опрацювання теорії:

* Тема №1: Робота з файлами в C++.
  + Джерела Інформації
* “The C++ Programming Language” by Bjarne Stroustrup
* “C++ For Dummies” by Stephen R. Davis
  + Що опрацьовано:
* Поняття списку
* Основні операції зі списками
* Приклади використання списків
* Реалізація списків мовою C++
  + Статус: Ознайомлений
  + Початок опрацювання теми: 30.11.2023
  + Звершення опрацювання теми: 12.12.2023
* Тема №2: Стек, черга
  + Джерела Інформації:
* “The C++ Programming Language” by Bjarne Stroustrup
* “C++ For Dummies” by Stephen R. Davis
  + Що опрацьовано:
* Поняття стека
* Основні операції зі стеком
* Поняття черги
* Основні операції з чергою
* Алгоритми обробки стеків і черг
* Реалізація черги та стеку мовою C++
  + Статус: Ознайомлений
  + Початок опрацювання теми: 30.11.2023
  + Звершення опрацювання теми: 12.12.2023
* Тема №3: Дерево
  + Джерела Інформації:
* “The C++ Programming Language” by Bjarne Stroustrup
* “C++ For Dummies” by Stephen R. Davis
  + Що опрацьовано:
* Поняття дерева
* Алгоритм запису даних у файлОсновні операції з деревами
* Алгоритми обробки дерев
* Реалізація дерева мовою C++
  + Статус: Ознайомлений
  + Початок опрацювання теми: 30.11.2023
  + Звершення опрацювання теми: 12.12.2023

# **Виконання роботи:**

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

Завдання **№1 VNS Lab 10 Task 1**

* 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. 22.Записи в лінійному списку містять ключове поле типу \*char (рядок символів). Сформувати двонаправлений список. Знищити елемент із заданим ключем. Додати К елементів перед елементом із заданим ключем.

Завдання **№2  Algotester Lab 5v3.**

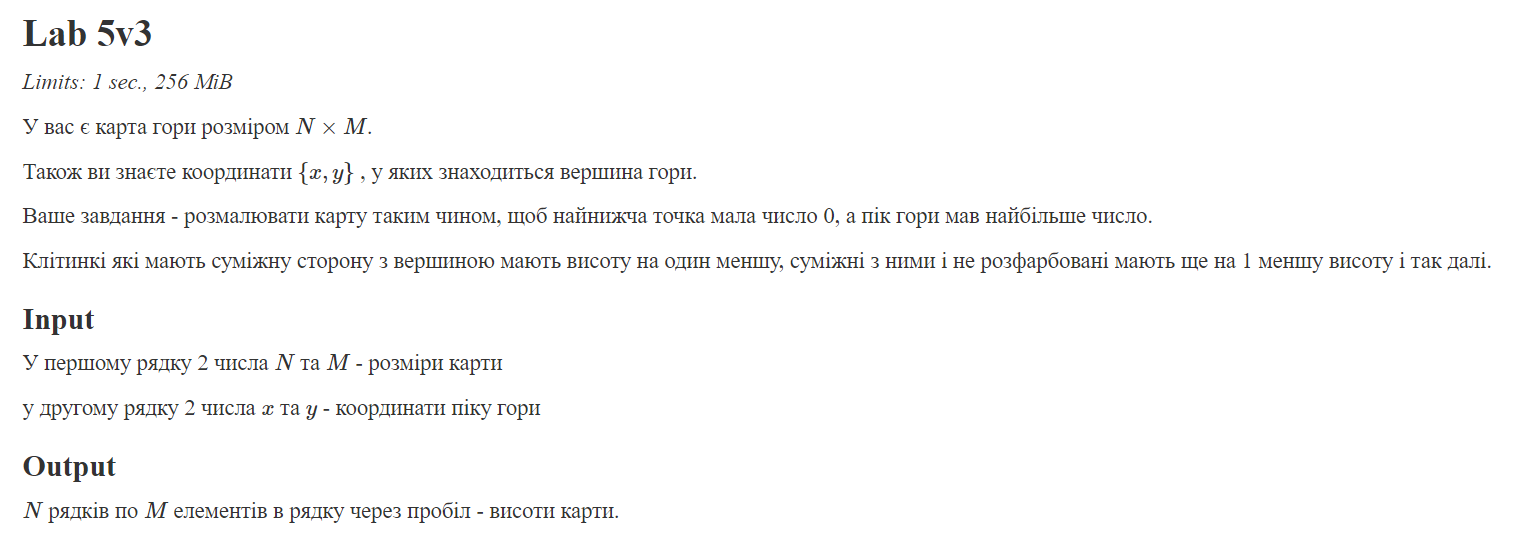


Рисунок 1 Завдання до Algotester Lab 5

Завдання **№3  Algotester Lab 78v3.**

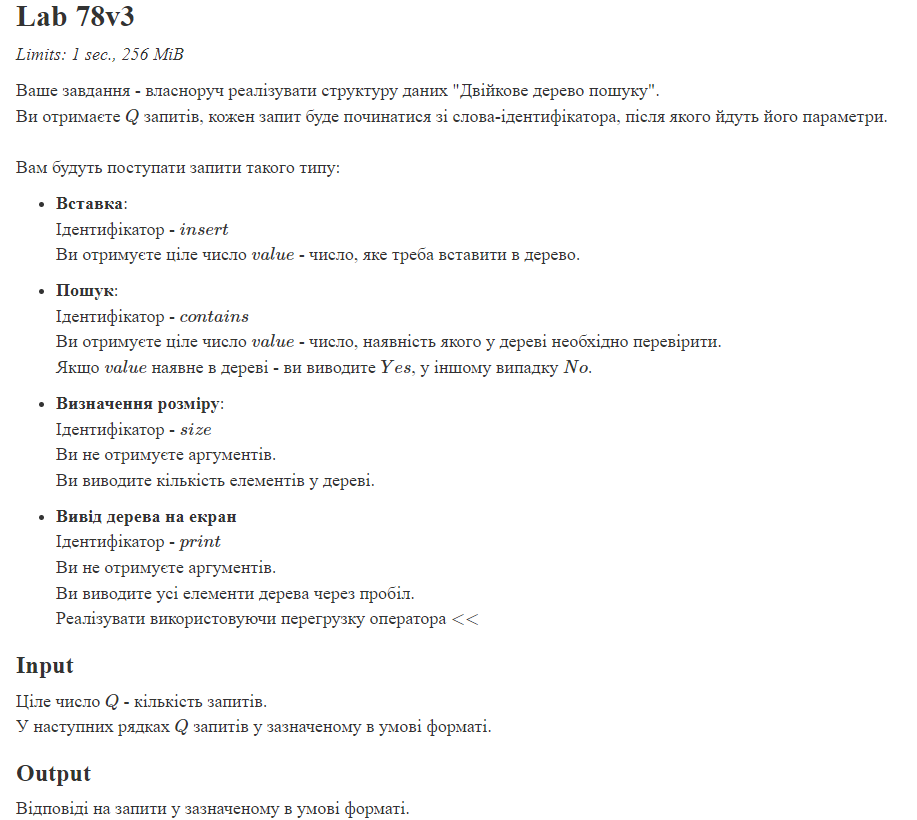


Рисунок 2 Завдання до Algotester Lab 78

Завдання **№4  Practice Work**

* ***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;

* 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***.

* 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 **№5 Self Practice Work**

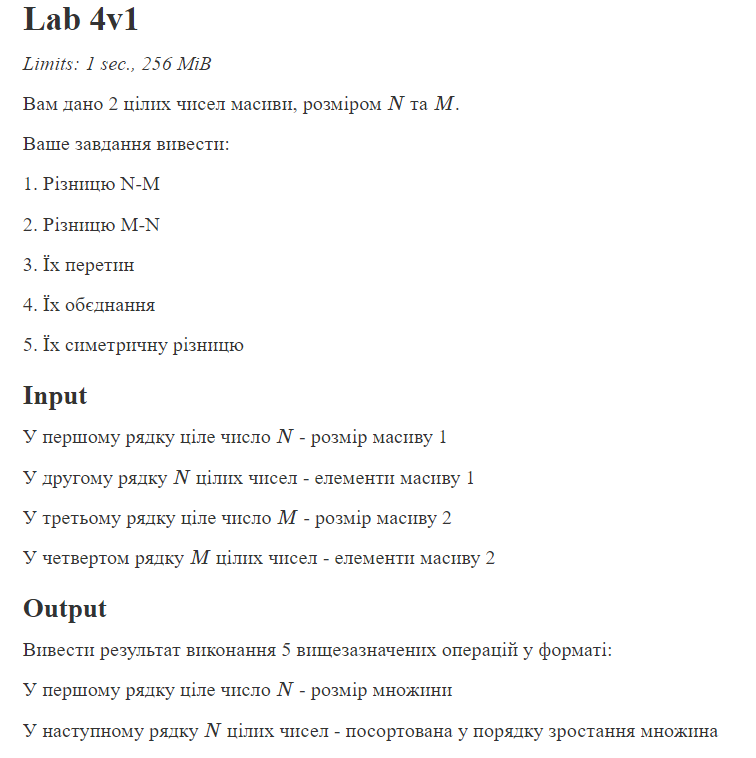


Рисунок 3 Завдання до Self Practice

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

Програма **№1 VNS Lab 10 Task 1**

* Планований час на реалізацію: **45 хвилин**.

Програма **№2 Algotester Lab 5 Task 1.**

* Блок-схема

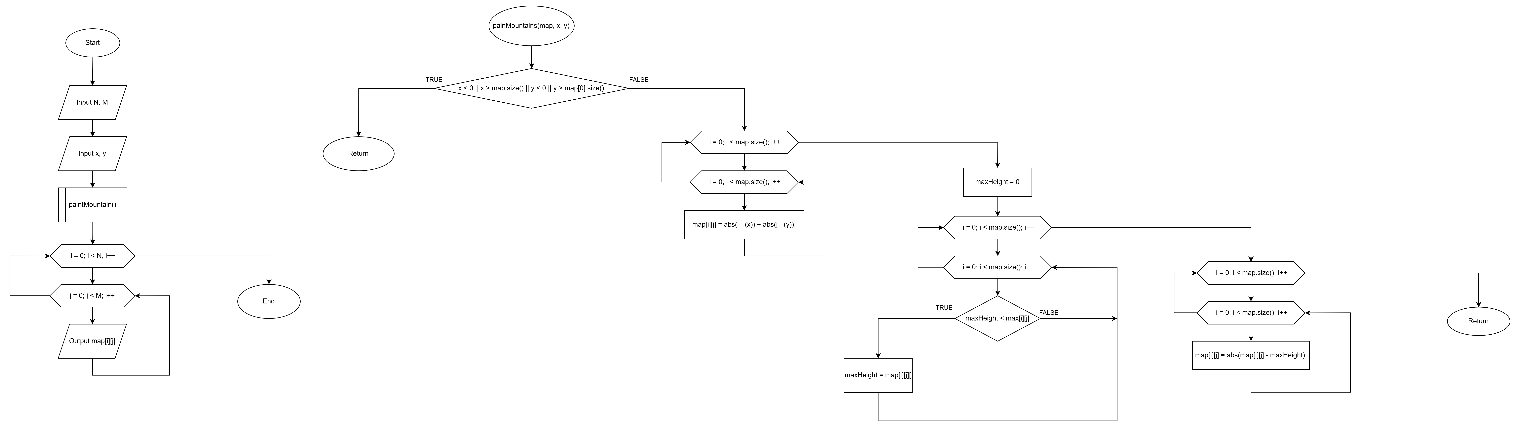


Рисунок 4 Блок схема до програми №3

* Планований час на реалізацію: **55 хвилин**.

Програма **№3 Algotester Lab 78 Task 1.**

* Блок-схема
* Планований час на реалізацію: **1.5 години**.

Програма **№4 Practice Work.**

* Планований час на реалізацію: **2 години**.

Програма **№5 Self Practice Work.**

* Планований час на реалізацію: **25 хвилин**.

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

Завдання **№1 VNS Lab 10 Task 1**

Деталі по програмі + Вставка з кодом з підписами до вставки

//  
// Created by olehio-p on 12/7/2023.  
//  
#include <iostream>  
  
struct Node {  
 int data;  
 Node\* prev;  
 Node\* next;  
  
 explicit Node(int value) {  
 data = value;  
 prev = nullptr;  
 next = nullptr;  
 }  
};  
  
class DoubleLinkedList {  
 private:  
 Node\* head;  
 Node\* tail;  
  
 public:  
 DoubleLinkedList() {  
 head = nullptr;  
 tail = nullptr;  
 }  
  
 void append(int value) {  
 Node\* newNode = new Node(value);  
 if (!head) {  
 head = tail = newNode;  
 } else {  
 tail->next = newNode;  
 newNode->prev = tail;  
 tail = newNode;  
 }  
 }  
  
 void deleteByKey(int key) {  
 Node\* current = head;  
 while (current) {  
 if (current->data == key) {  
 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;  
 break;  
 }  
 current = current->next;  
 }  
 }  
  
 void addBeforeKey(int key, int k) {  
 Node\* current = head;  
 while (current) {  
 if (current->data == key) {  
 for (int i = 0; i < k; ++i) {  
 Node\* newNode = new Node(0);  
 newNode->next = current;  
 newNode->prev = current->prev;  
  
 if (current->prev) {  
 current->prev->next = newNode;  
 } else {  
 head = newNode;  
 }  
  
 current->prev = newNode;  
 }  
 break;  
 }  
 current = current->next;  
 }  
 }  
  
 void display() {  
 Node\* current = head;  
 while (current) {  
 std::cout << current->data << " ";  
 current = current->next;  
 }  
 std::cout << std::endl;  
 }  
};  
  
int main() {  
 DoubleLinkedList list;  
  
 list.append(1);  
 list.append(2);  
 list.append(3);  
 list.append(4);  
 list.append(5);  
  
 std::cout << "List: ";  
 list.display();  
  
 list.deleteByKey(3);  
 std::cout << "List after deleting: ";  
 list.display();  
  
 list.addBeforeKey(4, 2);  
 std::cout << "List after adding: ";  
 list.display();  
  
 return 0;  
}

Рисунок 5 Код до програми №1

Посилання на файл програми у пул-запиті GitHub

<https://github.com/artificial-intelligence-department/ai_programming_playground/blob/epic_4_practice_and_labs_oleh_popovych/ai_11/oleh_popovych/epic_4/vns_lab_4_task_1_variant_22_oleh_popovych.cpp>

Завдання **№2 Algotester Lab 5 Task 1**

Деталі по програмі + Вставка з кодом з підписами до вставки

//  
// Created by olehio-h on 12/7/2023.  
//  
#include <iostream>  
#include <string>  
#include <vector>  
  
  
void paintMountain(std::vector<std::vector<int>>, int, int);  
  
int main() {  
 int N, M;  
 std::cin >> N >> M;  
  
 int x, y;  
 std::cin >> x >> y;  
 --x;  
 --y;  
  
 std::vector<std::vector<int>> map(N, std::vector<int>(M));  
  
 paintMountain(map, x, y);  
  
 for (int i = 0; i < N; i++) {  
 for (int j = 0; j < M; j++) {  
 std::cout << map[i][j] << " ";  
 }  
 std::cout << std::endl;  
 }  
  
 return 0;  
}  
  
void paintMountain(std::vector<std::vector<int>>& map, int x, int y) {  
 if (x < 0 || x > map.size() || y < 0 || y > map[0].size()) {  
 return;  
 }  
  
 for (int i = 0; i < map.size(); i++) {  
 for (int j = 0; j < map[i].size(); j++) {  
 map[i][j] = abs(i - (x)) + abs(j - (y));  
 }  
 }  
  
 int maxHeight = 0;  
 for (int i = 0; i < map.size(); i++) {  
 for (int j = 0; j < map[0].size(); j++) {  
 if (maxHeight < map[i][j])  
 maxHeight = map[i][j];  
 }  
 }  
  
 for (int i = 0; i < map.size(); i++) {  
 for (int j = 0; j < map[0].size(); j++) {  
 map[i][j] = abs(map[i][j] - maxHeight);  
 }  
 }  
}

Рисунок 8 Код до програми №4

Посилання на файл програми у пул-запиті GitHub

<https://github.com/artificial-intelligence-department/ai_programming_playground/blob/epic_4_practice_and_labs_oleh_popovych/ai_11/oleh_popovych/epic_4/vns_lab_4_task_1_variant_22_oleh_popovych.cpp>

Завдання **№3 Algotester Lab 78 Task 1**

Деталі по програмі + Вставка з кодом з підписами до вставки

//  
// Created by olehio-p on 12/7/2023.  
//  
#include <iostream>  
  
class BinarySearchTree {  
 private:  
 struct Node {  
 int data;  
 Node\* left;  
 Node\* right;  
  
 explicit Node(int value) {  
 data = value;  
 left = nullptr;  
 right = nullptr;  
 }  
 };  
  
 Node\* root;  
  
 Node\* insert0(Node\* current, int value) {  
 if (!current) {  
 return new Node(value);  
 }  
  
 if (value < current->data) {  
 current->left = insert0(current->left, value);  
 } else if (value > current->data) {  
 current->right = insert0(current->right, value);  
 }  
  
 return current;  
 }  
  
 bool contains0(Node\* current, int value) const {  
 if (!current) {  
 return false;  
 }  
  
 if (value == current->data) {  
 return true;  
 } else if (value < current->data) {  
 return contains0(current->left, value);  
 } else {  
 return contains0(current->right, value);  
 }  
 }  
  
 int size0(Node\* current) const {  
 if (!current) {  
 return 0;  
 }  
  
 return 1 + size0(current->left) + size0(current->right);  
 }  
  
 void print0(Node\* current) const {  
 if (current) {  
 print0(current->left);  
 std::cout << current->data << " ";  
 print0(current->right);  
 }  
 }  
  
 public:  
 BinarySearchTree() {  
 root = nullptr;  
 }  
  
 void insert(int value) {  
 root = insert0(root, value);  
 }  
  
 bool contains(int value) {  
 return contains0(root, value);  
 }  
  
 int size() {  
 return size0(root);  
 }  
  
 void print() {  
 print0(root);  
 std::cout << std::endl;  
 }  
};  
  
int main() {  
 int Q;  
 std::cin >> Q;  
  
 BinarySearchTree bst;  
  
 while (Q--) {  
 std::string identifier;  
 std::cin >> identifier;  
  
 if (identifier == "insert") {  
 int value;  
 std::cin >> value;  
 bst.insert(value);  
 } else if (identifier == "contains") {  
 int value;  
 std::cin >> value;  
 std::cout << (bst.contains(value) ? "Yes" : "No") << std::endl;  
 } else if (identifier == "size") {  
 std::cout << bst.size() << std::endl;  
 } else if (identifier == "print") {  
 bst.print();  
 }  
 }  
  
 return 0;  
}

Рисунок 9 Код до програми №5

Посилання на файл програми у пул-запиті GitHub

<https://github.com/artificial-intelligence-department/ai_programming_playground/blob/epic_4_practice_and_labs_oleh_popovych/ai_11/oleh_popovych/epic_4/vns_lab_4_task_1_variant_22_oleh_popovych.cpp>

Завдання **№4 Practice Work Task 1**

Деталі по програмі + Вставка з кодом з підписами до вставки

//  
// Created by olehio-p on 12/7/2023.  
//  
#include <iostream>  
  
struct Node {  
 int data;  
 Node\* next;  
  
 explicit Node(int val) {  
 data = val;  
 next = nullptr;  
 }  
};  
  
void append(Node\*& head, int val) {  
 Node\* newNode = new Node(val);  
 if (!head) {  
 head = newNode;  
 return;  
 }  
  
 Node\* temp = head;  
 while (temp->next) {  
 temp = temp->next;  
 }  
  
 temp->next = newNode;  
}  
  
void printList(Node\* head) {  
 while (head) {  
 std::cout << head->data << " ";  
 head = head->next;  
 }  
 std::cout << std::endl;  
}  
  
Node\* reverse(Node\* head) {  
 Node\* prev = nullptr;  
 Node\* current = head;  
 Node\* next = nullptr;  
  
 while (current) {  
 next = current->next;  
 current->next = prev;  
 prev = current;  
 current = next;  
 }  
  
 return prev;  
}  
  
bool compare(Node\* list1, Node\* list2) {  
 while (list1 && list2) {  
 if (list1->data != list2->data) {  
 return false;  
 }  
 list1 = list1->next;  
 list2 = list2->next;  
 }  
  
 return (list1 == nullptr && list2 == nullptr);  
}  
  
Node\* addLists(Node\* list1, Node\* list2) {  
 Node\* result = nullptr;  
 Node\* current = nullptr;  
 int carry = 0;  
  
 while (list1 || list2 || carry) {  
 int sum = (list1 ? list1->data : 0) + (list2 ? list2->data : 0) + carry;  
 carry = sum / 10;  
  
 if (result == nullptr) {  
 result = new Node(sum % 10);  
 current = result;  
 } else {  
 current->next = new Node(sum % 10);  
 current = current->next;  
 }  
  
 if (list1) list1 = list1->next;  
 if (list2) list2 = list2->next;  
 }  
  
 return result;  
}  
  
  
int main() {  
 Node\* head = nullptr;  
 append(head, 1);  
 append(head, 2);  
 append(head, 3);  
 append(head, 4);  
  
 std::cout << "Input List: ";  
 printList(head);  
  
 head = reverse(head);  
  
 std::cout << "Reversed List: ";  
 printList(head);  
  
 Node\* list1 = nullptr;  
 append(list1, 1);  
 append(list1, 2);  
 append(list1, 3);  
 std::cout << "List 1: ";  
 printList(list1);  
  
 Node\* list2 = nullptr;  
 append(list2, 1);  
 append(list2, 2);  
 append(list2, 3);  
 std::cout << "List 2: ";  
 printList(list2);  
  
 bool result = compare(list1, list2);  
  
 std::cout << "Lists are " << (result ? "equal" : "not equal") << std::endl;  
  
 Node\* list3 = nullptr;  
 append(list3, 9);  
 append(list3, 7);  
 append(list3, 3);  
  
 Node\* list4 = nullptr;  
 append(list4, 5);  
 append(list4, 6);  
 append(list4, 4);  
  
 Node\* sumList = addLists(list3, list4);  
  
 std::cout << "List 3: ";  
 list3 = reverse(list3);  
 printList(list3);  
  
 list4 = reverse(list4);  
 std::cout << "List 4: ";  
 printList(list4);  
  
 std::cout << "Sum List: ";  
 sumList = reverse(sumList);  
 printList(sumList);  
  
 return 0;  
}

Рисунок 10 Код до програми №6

Посилання на файл програми у пул-запиті GitHub

<https://github.com/artificial-intelligence-department/ai_programming_playground/blob/epic_4_practice_and_labs_oleh_popovych/ai_11/oleh_popovych/epic_4/vns_lab_4_task_1_variant_22_oleh_popovych.cpp>

Завдання **№5 Self Practice Task 1**

Деталі по програмі + Вставка з кодом з підписами до вставки

//  
// Created by olehio-p on 12/7/2023.  
//  
#include <iostream>  
#include <vector>  
#include <algorithm>  
  
int main() {  
 int N;  
 std::cin >> N;  
 std::vector<int> array1(N);  
 for (int i = 0; i < N; ++i) {  
 std::cin >> array1[i];  
 }  
  
 int M;  
 std::cin >> M;  
 std::vector<int> array2(M);  
 for (int i = 0; i < M; ++i) {  
 std::cin >> array2[i];  
 }  
  
 std::sort(array1.begin(), array1.end());  
 std::sort(array2.begin(), array2.end());  
  
 std::vector<int> differenceNM;  
 std::set\_difference(array1.begin(), array1.end(), array2.begin(), array2.end(), std::back\_inserter(differenceNM));  
  
 std::vector<int> differenceMN;  
 std::set\_difference(array2.begin(), array2.end(), array1.begin(), array1.end(), std::back\_inserter(differenceMN));  
  
 std::vector<int> intersection;  
 std::set\_intersection(array1.begin(), array1.end(), array2.begin(), array2.end(), std::back\_inserter(intersection));  
  
 std::vector<int> unionSet;  
 std::set\_union(array1.begin(), array1.end(), array2.begin(), array2.end(), std::back\_inserter(unionSet));  
  
 std::vector<int> symmetricDifference;  
 std::set\_symmetric\_difference(array1.begin(), array1.end(), array2.begin(), array2.end(), std::back\_inserter(symmetricDifference));  
  
 std::cout << differenceNM.size() << std::endl;  
 for (const auto& elem : differenceNM) {  
 std::cout << elem << " ";  
 }  
 std::cout << std::endl;  
  
 std::cout << differenceMN.size() << std::endl;  
 for (const auto& elem : differenceMN) {  
 std::cout << elem << " ";  
 }  
 std::cout << std::endl;  
  
 std::cout << intersection.size() << std::endl;  
 for (const auto& elem : intersection) {  
 std::cout << elem << " ";  
 }  
 std::cout << std::endl;  
  
 std::cout << unionSet.size() << std::endl;  
 for (const auto& elem : unionSet) {  
 std::cout << elem << " ";  
 }  
 std::cout << std::endl;  
  
 std::cout << symmetricDifference.size() << std::endl;  
 for (const auto& elem : symmetricDifference) {  
 std::cout << elem << " ";  
 }  
 std::cout << std::endl;  
  
 return 0;  
}

Рисунок 11 Код до програми №7

Посилання на файл програми у пул-запиті GitHub

<https://github.com/artificial-intelligence-department/ai_programming_playground/blob/epic_4_practice_and_labs_oleh_popovych/ai_11/oleh_popovych/epic_4/vns_lab_4_task_1_variant_22_oleh_popovych.cpp>

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

Завдання **№1 VNS Lab 10 Task 1**

Деталі виконання і тестуванню програми

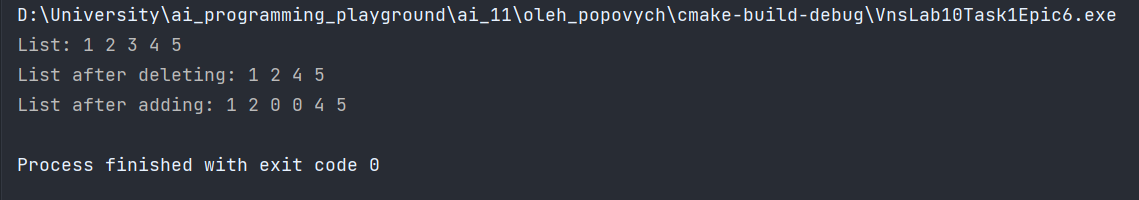


Рисунок 12 Результат виконання програми №1

Час затрачений на виконання завдання: **45 хвилин**.

Завдання **№2  Algotester Lab 5 Task 1**

Деталі виконання і тестуванню програми

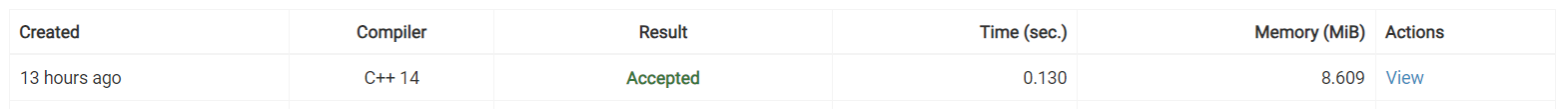


Рисунок 15 Результат виконання програми №4

Час затрачений на виконання завдання: **50 хвилин.**

Завдання **№3  Algotester Lab 78 Task 1**

Деталі виконання і тестуванню програми



Рисунок 16 Результат виконання програми №5

Час затрачений на виконання завдання: **1.5 хвилин.**

Завдання **№4  Practice Work.**

Деталі виконання і тестуванню програми

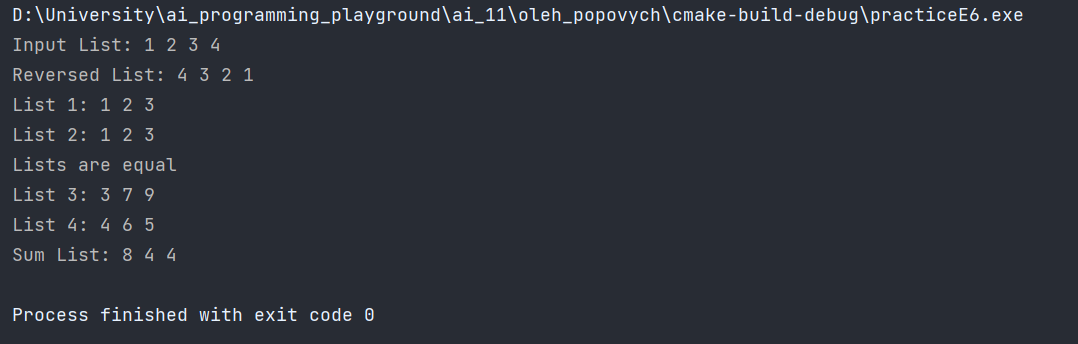


Рисунок 17 Результат виконання програми №6

Час затрачений на виконання завдання: **1.5 хвилин.**

Завдання **№5  Self Practice Work.**

Деталі виконання і тестуванню програми

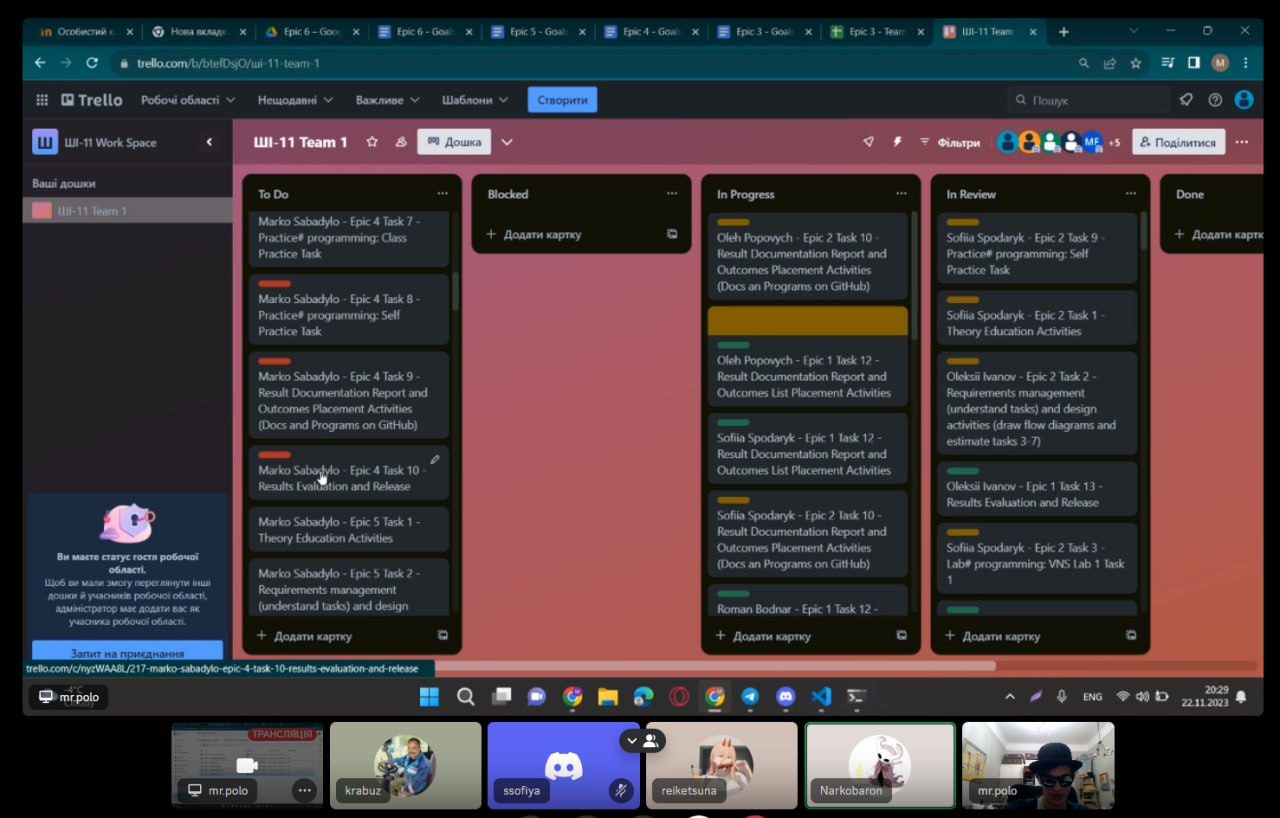


Рисунок 18 Результат виконання програми №7

Час затрачений на виконання завдання: **25 хвилин.**

## **6. Кооперація з командою:**

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



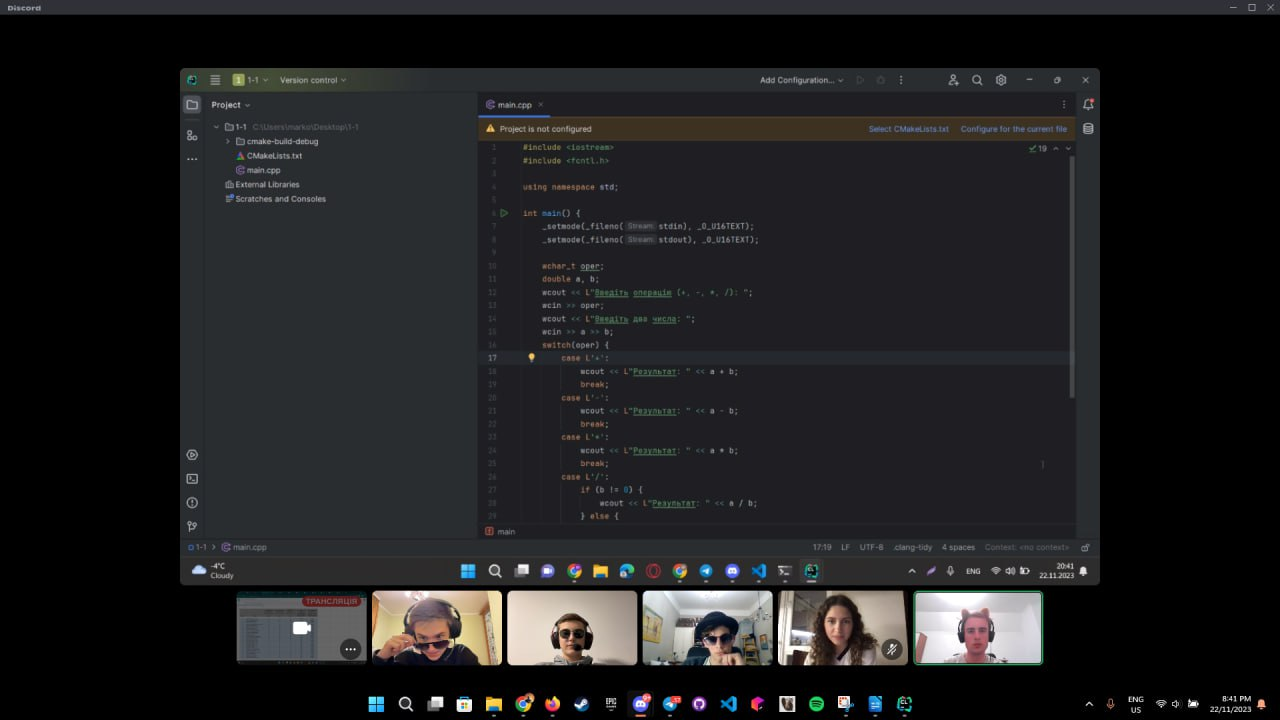


Рисунок 14 Cкріни зустрічі по обговоренню задач Епіку 4

# **Висновки:**

Ознайомився з динамічними структурами (Черга, Стек, Списки, Дерево). Алгоритми обробки динамічних структур

Виконав задані лабораторні роботи.

Виконав задане практичне завдання.