Міністерство освіти і науки України

Національний університет «Львівська політехніка»

Кафедра систем штучного інтелекту



**Звіт**

про виконання

**Лабораторних та практичних робіт № 6**

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

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

***Виконав(ла):***

студент групи ШІ-13

замінити на ПІБ повністю

# **Тема роботи:**

Ознайомлення з динамічними структурами, з чергою, стеком, списком та з бінарним деревом. Зрозуміти як це працює і використати знання на практиці

# **Мета роботи:**

Навчитись користуватись з динамічними структурами, бінарним деревом на мові С++. Виконати всі поставленні завдання закріпивши матеріал.

# **Теоретичні відомості:**

1. Теоретичні відомості з переліком важливих тем:

* Тема №1: Вивчення структур
* Тема №2: Бінарне дерево.
* Тема №3: Зв’язний список.
* Тема №4: Конструктори, деконструктори, вказівник this.

1. Індивідуальний план опрацювання теорії:

* Тема №1: Вивчення структур.
  + Джерела Інформації
    - Відео: https://www.youtube.com/watch?v=EWJqZN7ifI0
    - Стаття: https://acode.com.ua/urok-64-struktury/
  + Що опрацьовано:
    - Ознайомився з структурами, зрозумів як їх створювати
  + Статус: Ознайомлений
  + Початок опрацювання теми: 18.12.2023
  + Звершення опрацювання теми: 20.12.2023
* Тема №2: Бінарне дерево.
  + Джерела Інформації:
    - Відео : https://www.youtube.com/watch?v=9o\_i0zzxk1s
    - Стаття : https://uk.myservername.com/binary-search-tree-c
  + Що опрацьовано:
    - Ознайомився з бінарним деревом, зрозумів як його використовувати на практиці.
  + Статус: Ознайомлений
  + Початок опрацювання теми: 18.12.2023
  + Звершення опрацювання теми: 20.12.2023
* Тема №3: Зв’язний список.
  + Джерела Інформації:
    - Відео.
    - Стаття : https://www.bestprog.net/uk/2022/02/11/c-linear-singly-linked-list-general-information-ua/
  + Що опрацьовано:
    - Ознайомився зі зв’язним списоком та використав його на практиці.
  + Статус: Ознайомлений
  + Початок опрацювання теми: 18.12.2023
  + Звершення опрацювання теми: 20.12.2023
* Тема №4: Конструктори, деконструктори, вказівник this.
  + Джерела Інформації:
    - Відео: https://www.youtube.com/watch?v=3QcSjhJHA70
    - Стаття: https://purecodecpp.com/archives/1792
  + Що опрацьовано:
  + Навчився користуватись вказівником this, ознайомився з конструктори та деконструктори.
  + Статус: Ознайомлений
  + Початок опрацювання теми: 18.12.2023
  + Звершення опрацювання теми: 20.12.2023

# **Виконання роботи:**

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

Завдання №1 **VNS Lab 10**

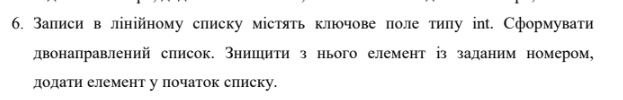
* Варіант завдання: 6
* Деталі завдання 

Рисунок Пояснення до завдання VnsLab10

* Важливі деталі для врахування в імплементації програми: -

Завдання №2 **Algotester Lab 5**

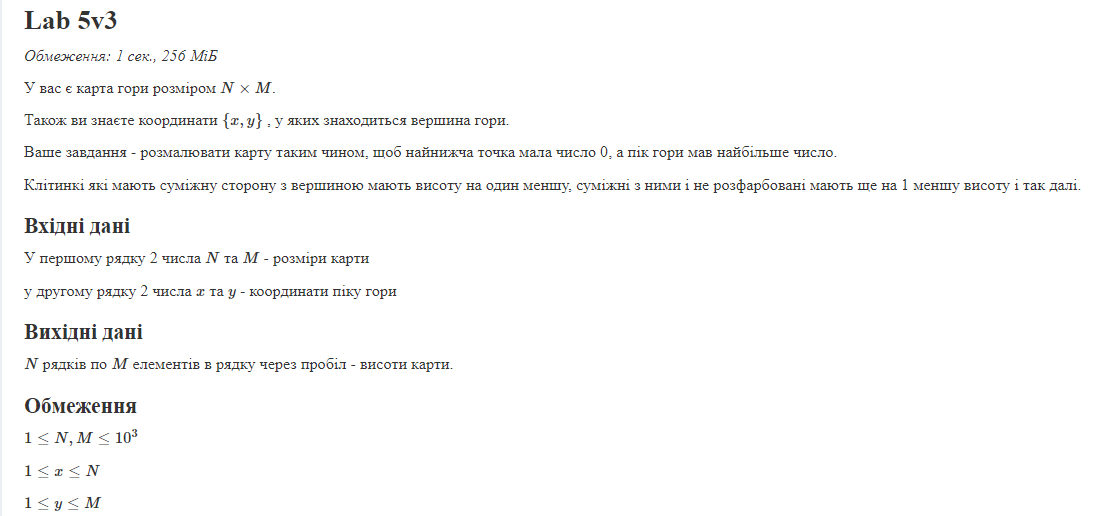
* Варіант завдання: 3
* Деталі завдання 

Рисунок Пояснення до завдання AlgoLab5v3

* Важливі деталі для врахування в імплементації програми: -

Завдання №3 **Algotester Lab 7-8**

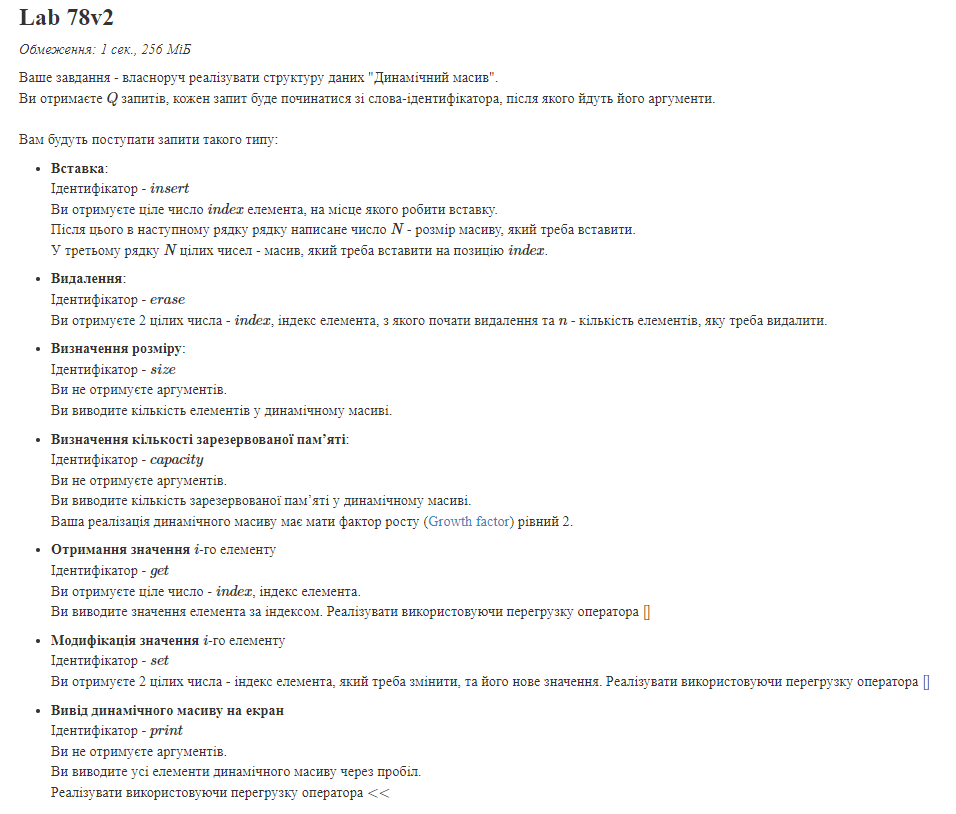
* Варіант завдання: 2
* Деталі завдання: 

Рисунок Пояснення до завдання AlgoLab78v2

* Важливі деталі для врахування в імплементації програми:-

Завдання №4 **Class Practice Work-1**

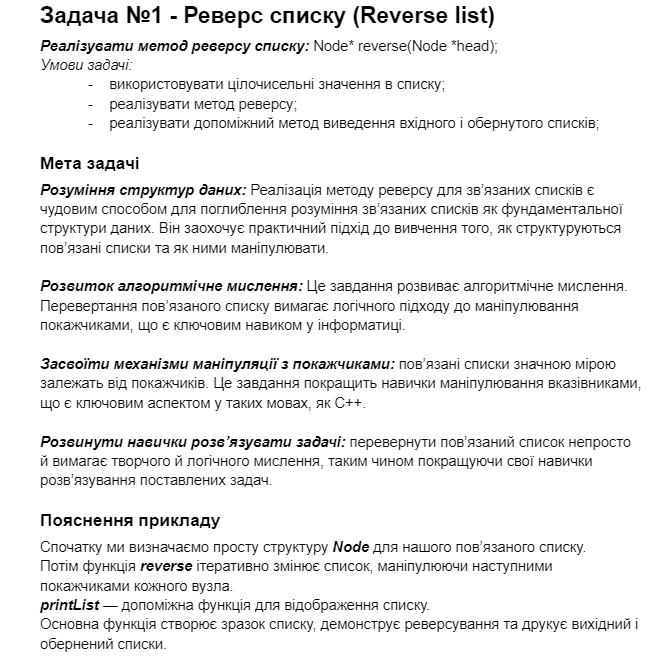
* Варіант завдання: 1
* Деталі завдання :
* 

Рисунок Пояснення до завдання Practice-work-1

* Важливі деталі для врахування в імплементації програми:-

Завдання №5 **Class Practice Work-2**

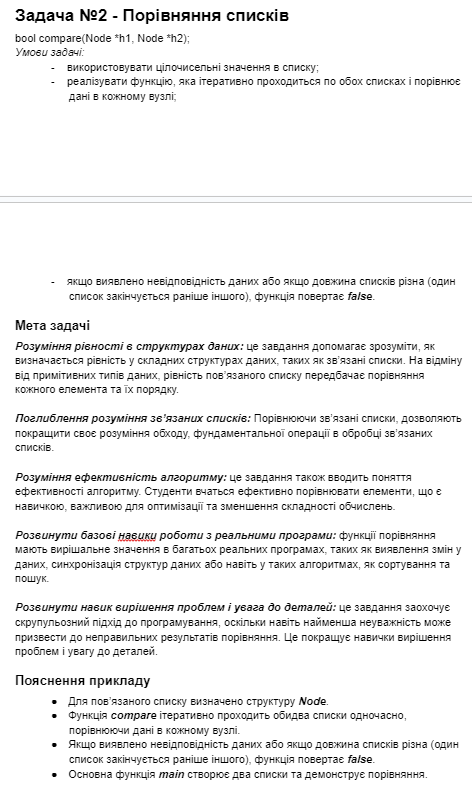
* Варіант завдання: 2
* Деталі завдання:
* 

Рисунок Пояснення до завдання Practice-work-2

* Важливі деталі для врахування в імплементації програми

Завдання №6 **Class Practice Work-3** заголовок завдання

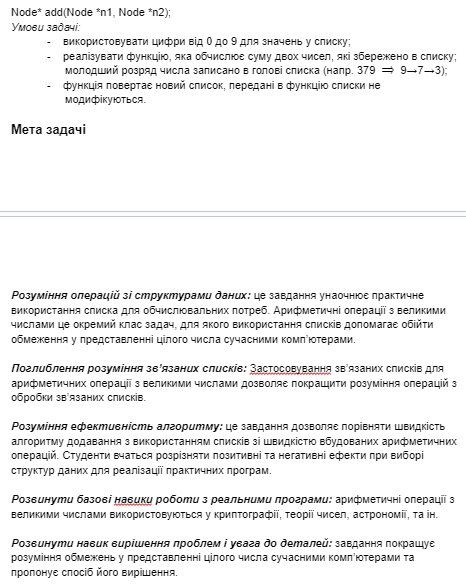
* Варіант завдання: 3
* Деталі завдання:  
  

Рисунок Пояснення до завдання Practice-work-3

* Важливі деталі для врахування в імплементації програми

Завдання №7 **Class Practice Work-4** заголовок завдання

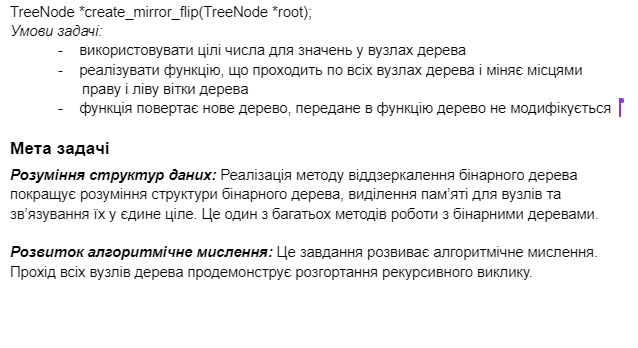
* Варіант завдання: 4
* Деталі завдання:
* 

Рисунок Пояснення до завдання Practice-work-4

* Важливі деталі для врахування в імплементації програми

Завдання №8 **Class Practice Work-5** заголовок завдання

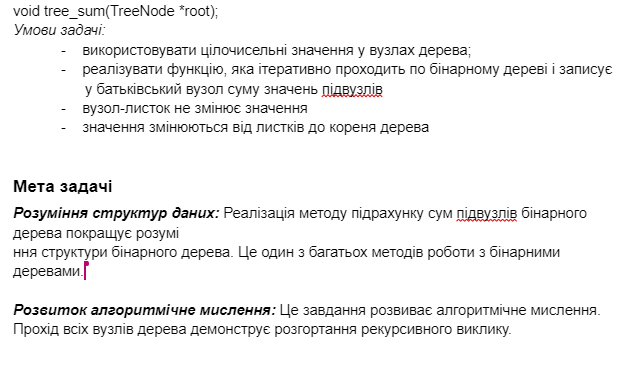
* Варіант завдання: 5
* Деталі завдання:
* 

Рисунок Пояснення до завдання Practice-work-5

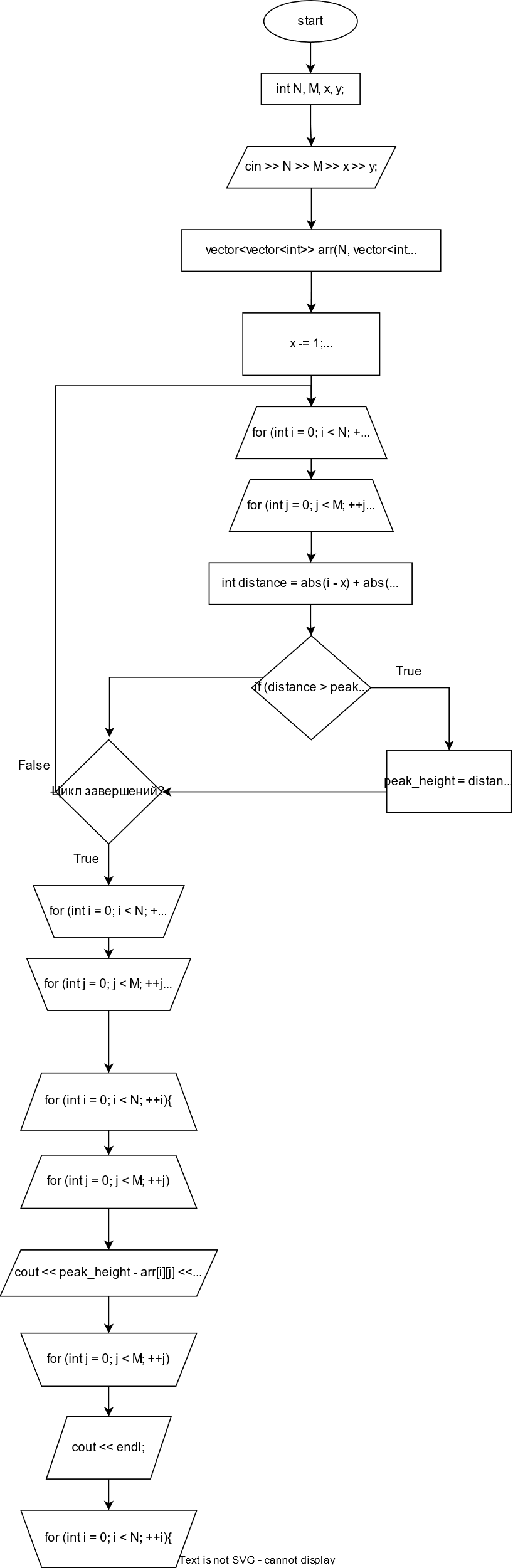
* Важливі деталі для врахування в імплементації програми

Завдання №9 **Self Practice Work** заголовок завдання

* Варіант завдання
* Деталі завдання
* Важливі деталі для врахування в імплементації програми

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

Програма №1 Algo5v3

* Блок-схема
* 

1 Блок схема до завдання Algo5v3

* Планований час на реалізацію: 30 хв
* Важливі деталі для врахування в імплементації: -

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

ПР - https://github.com/artificial-intelligence-department/ai\_programming\_playground/pull/1225/commits/1b092fa422665ae8e01a7803806268b3bcc4a0bb

Завдання №1 VnsLab10v6

#include <iostream>

using namespace std;

struct Node {

int key;

Node\* prev;

Node\* next;

Node(int val) : key(val), prev(nullptr), next(nullptr) {}

};

class DoublyLinkedList {

Node\* head;

public:

DoublyLinkedList() : head(nullptr) {}

void addToFront(int key) {

Node\* newNode = new Node(key);

if (head == nullptr)

head = newNode;

else {

newNode->next = head;

head->prev = newNode;

head = newNode;

}

}

void deleteNode(int position) {

if (head == nullptr) {

cout << "empty list" << endl;

return;

}

Node\* current = head;

int currentPosition = 1;

while (current != nullptr && currentPosition < position) {

current = current->next;

currentPosition++;

}

if (current == nullptr) {

cout << "element wasn`t found." << endl;

return;

}

if (current->prev != nullptr)

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

else

head = current->next;

if (current->next != nullptr)

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

delete current;

cout << "element number of " << position << " deleted" << endl;

}

void displayList() {

Node\* current = head;

while (current != nullptr) {

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

current = current->next;

}

cout << endl;

}

};

int main() {

DoublyLinkedList list;

list.addToFront(3);

list.addToFront(5);

list.addToFront(7);

list.addToFront(9);

cout << "list: ";

list.displayList();

list.deleteNode(2);

list.addToFront(1);

cout << "list: ";

list.displayList();

return 0;

}

Завдання №2 Algo5v3

#include <iostream>

#include <vector>

using namespace std;

int main()

{

int N, M, x, y;

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

vector<vector<int>> arr(N, vector<int>(M));

x -= 1;

y -= 1;

int peak\_height = 0;

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

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

int distance = abs(i - x) + abs(j - y);

arr[i][j] = distance;

if (distance > peak\_height)

peak\_height = distance;

}

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

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

cout << peak\_height - arr[i][j] << " ";

cout << endl;

}

return 0;

}

Завдання №3 Algo78v2

#include <iostream>

using namespace std;

class DynArr {

private:

int\* array;

int currSize;

int currCap;

void resize(int newCapacity) {

int\* tempArray = new int[newCapacity];

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

tempArray[i] = array[i];

}

delete[] array;

array = tempArray;

currCap = newCapacity;

}

public:

DynArr() : array(nullptr), currSize(0), currCap(1) {

array = new int[currCap];

}

~DynArr() {

delete[] array;

}

void insert(int index, int N, int\* elements) {

if (currSize + N >= currCap) {

while (currSize + N >= currCap)

currCap \*= 2;

resize(currCap);

}

for (int i = currSize - 1; i >= index; --i)

array[i + N] = array[i];

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

array[index + i] = elements[i];

currSize += N;

}void erase(int index, int n) {

for (int i = index; i < currSize - n; ++i)

array[i] = array[i + n];

currSize -= n;

}

int capacity() {

return currCap;

}

int get(int index) {

return array[index];

}

int size() {

return currSize;

}

void set(int index, int value) {

array[index] = value;

}

int& operator[](int index) {

return array[index];

}

friend ostream& operator<<(ostream& os, const DynArr& darr) {

for (int i = 0; i < darr.currSize; ++i)

os << darr.array[i] << " ";

return os;

}

};

int main() {

int Q;

cin >> Q;

DynArr dynArr;

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

string queue;

cin >> queue;

if (queue == "insert") {

int index, N;

cin >> index >> N;

int\* elements = new int[N];

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

cin >> elements[j];

}

dynArr.insert(index, N, elements);

delete[] elements;

}

else if (queue == "erase") {

int index, n;

cin >> index >> n;

dynArr.erase(index, n);

}

else if (queue == "get") {

int index;

cin >> index;

cout << dynArr.get(index) << endl;

}

else if (queue == "set") {

int index, value;

cin >> index >> value;

dynArr.set(index, value);

}

else if (queue == "capacity")

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

else if (queue == "size")

cout << dynArr.size() << endl;

else if (queue == "print")

cout << dynArr << endl;

}

return 0;

}

Завдання №4 Practice-work-1

#include <iostream>

using namespace std;

struct Node {

int data;

Node\* next;

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

};

Node\* reverse(Node\* head) {

Node\* prev = nullptr;

Node\* current = head;

Node\* nextNode = nullptr;

while (current != nullptr) {

nextNode = current->next;

current->next = prev;

prev = current;

current = nextNode;

}

return prev;

}

void printList(Node\* head) {

while (head != nullptr) {

cout << head->data << " ";

head = head->next;

}

cout << 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);

cout << "original list: ";

printList(head);

head = reverse(head);

cout << "oberneniy list: ";

printList(head);

while (head != nullptr) {

Node\* temp = head;

head = head->next;

delete temp;

}

return 0;

}

Завдання №5 Practice-work-2

#include <iostream>

using namespace std;

struct Node {

int data;

Node\* next;

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

};

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

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 not rivni." << endl;

else

cout << "lists is rivni." << endl;

delete head2;

delete head1;

return 0;

}

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

{

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

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

return false;

h1 = h1->next;

h2 = h2->next;

}

if (h1 != nullptr || h2 != nullptr)

return false;

return true;

}

Завдання №6 Practice-work-3

#include <iostream>

using namespace std;

struct Node {

int data;

Node\* next;

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

};

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

Node\* result = nullptr;

Node\* current = nullptr;

int carry = 0;

while (n1 != nullptr || n2 != nullptr) {

int num1 = (n1 != nullptr) ? n1->data : 0;

int num2 = (n2 != nullptr) ? n2->data : 0;

int sum = num1 + num2 + carry;

carry = sum / 10;

int value = sum % 10;

Node\* newNode = new Node(value);

if (result == nullptr) {

result = newNode;

current = newNode;

}

else {

current->next = newNode;

current = newNode;

}

if (n1 != nullptr) n1 = n1->next;

if (n2 != nullptr) n2 = n2->next;

}

if (carry > 0) {

Node\* newNode = new Node(carry);

current->next = newNode;

}

return result;

}

void printList(Node\* head) {

while (head != nullptr) {

cout << head->data << " ";

head = head->next;

}

cout << endl;

}

int main() {

Node\* num1 = new Node(3);

num1->next = new Node(7);

num1->next->next = new Node(9);

Node\* num2 = new Node(4);

num2->next = new Node(5);

num2->next->next = new Node(6);

Node\* result = add(num1, num2);

cout << "result dodavania: ";

printList(result);

delete result;

delete num1;

delete num2;

return 0;

}

Завдання №7 Practice-work-4

#include <iostream>

using namespace std;

struct TreeNode {

int data;

TreeNode\* left;

TreeNode\* right;

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

};

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

if (root == nullptr)

return nullptr;

TreeNode\* mirroredLeft = create\_mirror\_flip(root->right);

TreeNode\* mirroredRight = create\_mirror\_flip(root->left);

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

mirroredRoot->left = mirroredLeft;

mirroredRoot->right = mirroredRight;

return mirroredRoot;

}

void printTree(TreeNode\* root) {

if (root != nullptr) {

printTree(root->left);

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

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);

root->left->right = new TreeNode(5);

root->right->left = new TreeNode(6);

root->right->right = new TreeNode(7);

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

cout << "start tree: ";

printTree(root);

cout << endl;

cout << "mirored tree: ";

printTree(mirroredTree);

delete mirroredTree;

delete root;

return 0;

}

Завдання №8 Practice-work-5

#include <iostream>

#include <stack>

using namespace std;

struct TreeNode {

int data;

TreeNode\* left;

TreeNode\* right;

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

};

void tree\_sum(TreeNode\* root) {

if (root == nullptr)

return;

stack<TreeNode\*> nodeStack;

TreeNode\* current = root;

TreeNode\* prev = nullptr;

while (current != nullptr || !nodeStack.empty()) {

while (current != nullptr) {

nodeStack.push(current);

current = current->left;

}

current = nodeStack.top();

if (current->right != nullptr && current->right != prev)

current->data += current->right->data;

nodeStack.pop();

prev = current;

current = nullptr;

}

}

void printTree(TreeNode\* root) {

if (root != nullptr) {

printTree(root->left);

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

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);

root->left->right = new TreeNode(5);

root->right->left = new TreeNode(6);

root->right->right = new TreeNode(7);

tree\_sum(root);

cout << "sum nodes of binar tree: ";

printTree(root);

cout << endl;

delete root;

return 0;

}

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

Завдання №1 Algo5v3

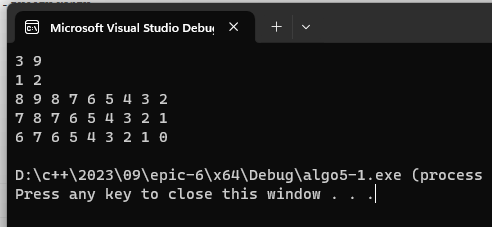


Figure Тестування програми Algo 5v3

Завдання №2 78v2

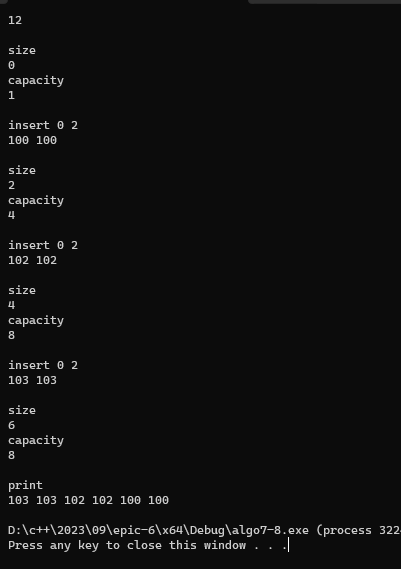


Figure Тестування програми Algo78v2

Завдання №3 Practice-work-1

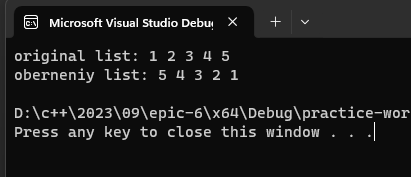


Figure Тестування програми Practice-work-1

Завдання №4 Practice-work-2

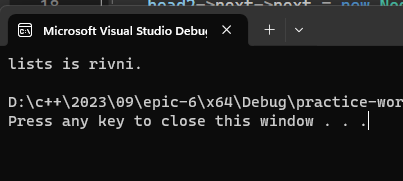


Figure Тестування програми Practice-work-2

Завдання №5 Practice-work-3

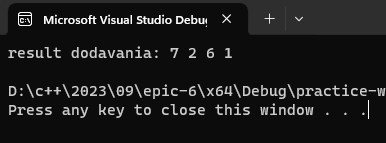


Figure Тестування програми Practice-work-3

Завдання №6 Practice-work-4

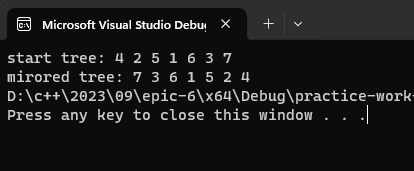


Figure Тестування програми Practice-work-4

Завдання №7 Practice-work-5

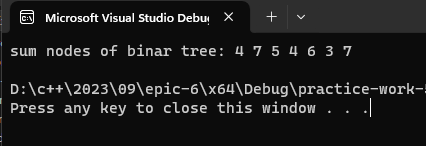
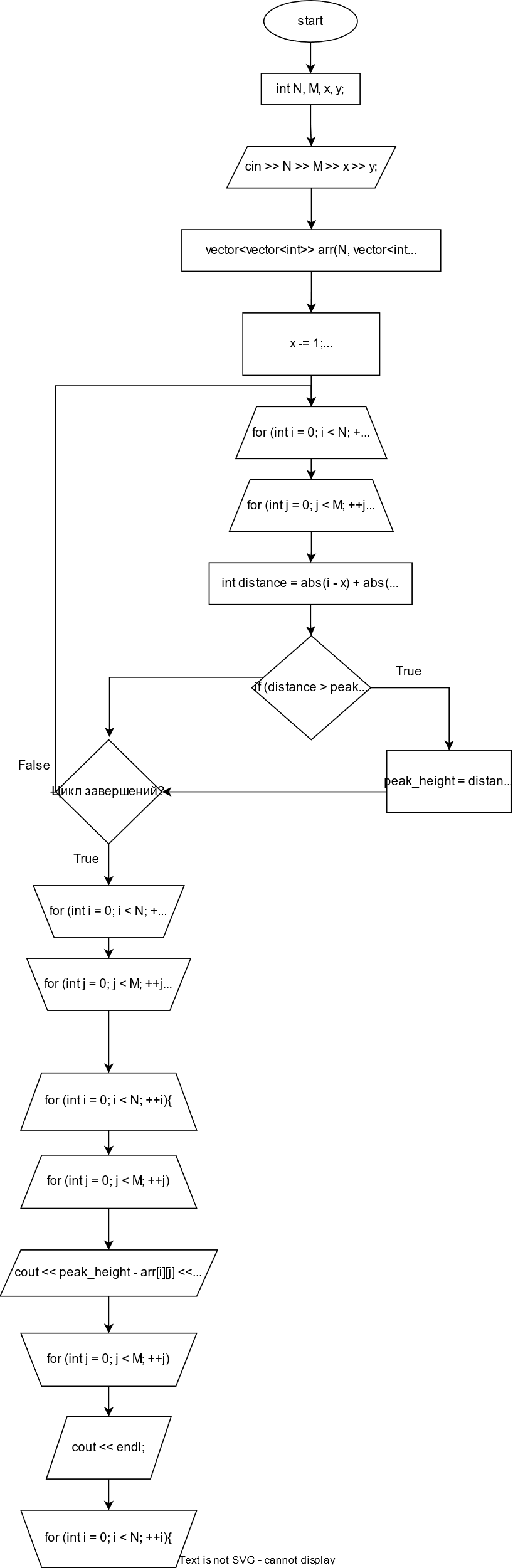
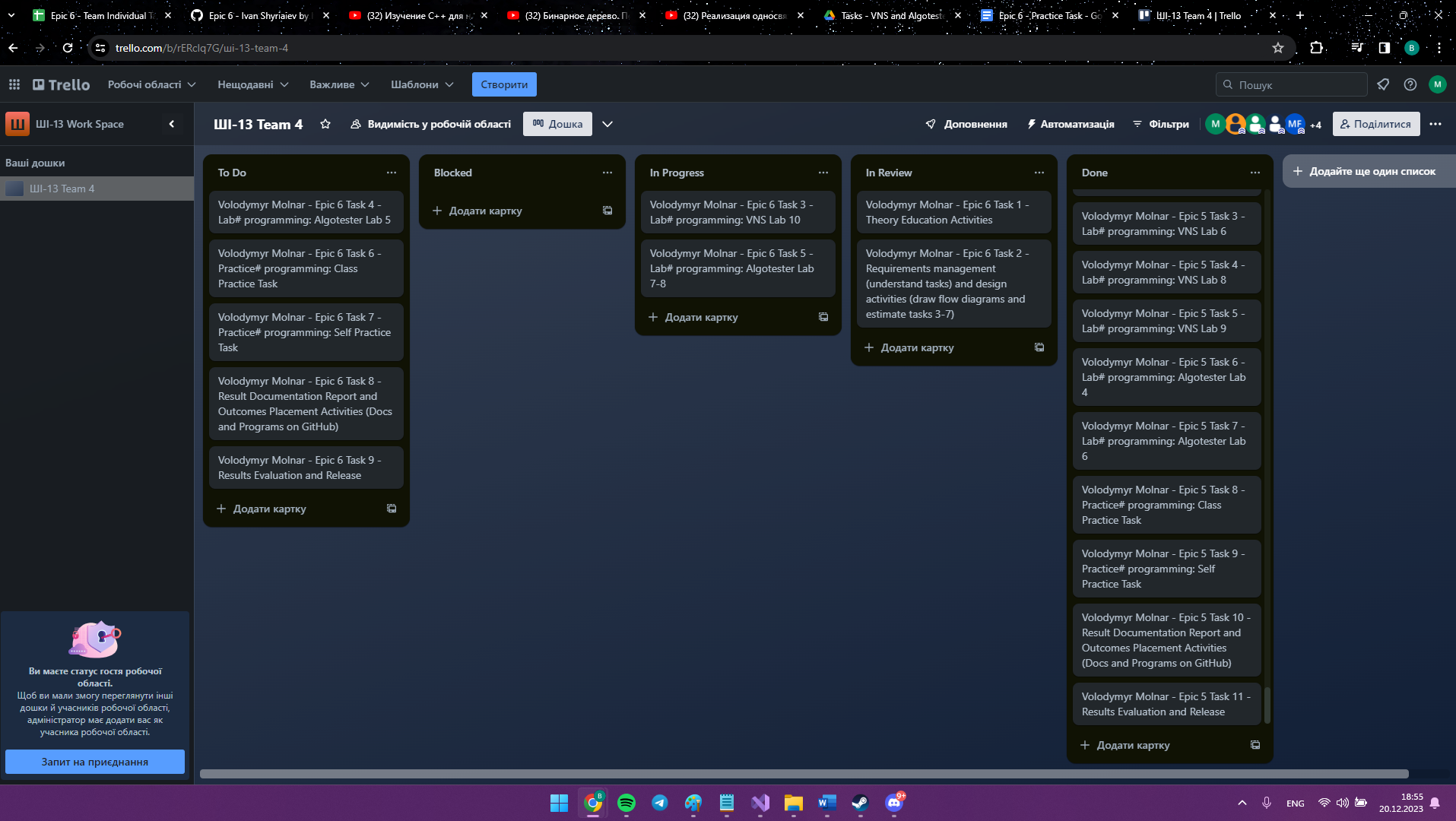


Figure Тестування програми Practice-work-5 

Час затрачений на виконання завдання

## **5. Кооперація з командою:**

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

# **Висновки:**

На цій лабораторній я вивчив як працювати з бінарним деревом. Зрозумів його переваги та навчився користуватись ним. Також навчився створювати структури і використовувати їх.