Міністерство освіти і науки України

Національний університет «Львівська політехніка»

Кафедра систем штучного інтелекту



**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-12

Яцишин Ігор Васильович

# **Тема роботи:**

Ознайомлення та робота з динамічними структурами даних в мові програмування C++. Принцип роботи з бінарними деревами. Ознайомлення з поняттями Стеку, Черги та Зв’язного Списку. Ознайомлення з основними алгоритмами обробки динамічних структур.

# **Мета роботи:**

Навчитися працювати з динамічними структурами даних в мові C++. Вивчити основні алгоритми обробки динамічних структур (Черга, Стек …). Застосувати набуті знання на практиці.

# **Теоретичні відомості:**

1. Теоретичні відомості з переліком важливих тем:

* Тема №1: Черга в C++.
* Тема №2: Стек в C++.
* Тема №3: Однозв’язний список в C++.
* Тема №4: Двозв’язний список в C++.
* Тема №5: Бінарні дерева пошуку в C++.

1. Індивідуальний план опрацювання теорії:

* Тема №1: Черга в C++.
  + Джерела Інформації
    - Стаття: Queue in C++ STL [Queue in C++ Standard Template Library (STL) - GeeksforGeeks](https://www.geeksforgeeks.org/queue-cpp-stl/)
    - Відео: C++ Теорія ⦁ Урок 142 ⦁ ADT ⦁ Черга [C++ Теорія ⦁ Урок 142 ⦁ ADT ⦁ Черга](https://www.youtube.com/watch?v=Yhw8NbjrSFA&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:
    - Опрацьовано теоретичний матеріал про роботу Черги в мові C++ та його застосування
  + Статус: Ознайомлений
  + Початок опрацювання теми: 07.12.2023 23.30
  + Звершення опрацювання теми: 08.11.2023 01.00
* Тема №2: Стек в C++.
  + Джерела Інформації
    - Стаття: Stack in C++ STL [Stack in C++ STL - GeeksforGeeks](https://www.geeksforgeeks.org/stack-in-cpp-stl/)
    - Відео: C++ Теорія ⦁ Урок 141 ⦁ ADT ⦁ Стек <https://www.youtube.com/watch?v=ZYvYISxaNL0&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:
    - Опрацьовано теоретичний матеріал про роботу Стеку в мові C++ та його застосування
  + Статус: Ознайомлений
  + Початок опрацювання теми: 07.12.2023 21.20
  + Звершення опрацювання теми: 07.11.2023 23.00
* Тема №3: Однозв’язний список в C++.
  + Джерела Інформації:
    - Відео: C++ Теорія ⦁ Урок 139 ⦁ Однозв’язний список [C++ Теорія ⦁ Урок 139 ⦁ ADT ⦁ Однозв'язний список](https://www.youtube.com/watch?v=-25REjF_atI&t=780s&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)
    - Стаття: Linked List Data Structure [Linked List Data Structure - GeeksforGeeks](https://www.geeksforgeeks.org/data-structures/linked-list/)
  + Що опрацьовано:
    - Опрацьовано поняття Однозв’язного списку в мові C++ та його використання
  + Статус: Ознайомлений
  + Початок опрацювання теми: 08.12.2023 22.30
  + Звершення опрацювання теми: 08.12.2023 23.50
* Тема №4: Двозв’язний список в C++.
  + Джерела Інформації:
    - Стаття: Doubly Linked List [Doubly Linked List (With code)](https://www.programiz.com/dsa/doubly-linked-list)
    - Відео: C++ Теорія ⦁ Урок 140 ⦁ Двозв’язний список [C++ Теорія ⦁ Урок 140 ⦁ ADT ⦁ Двозв'язний список](https://www.youtube.com/watch?v=QLzu2-_QFoE&t=4s&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:
    - Опрацьовано та вивчено основні методи роботи з Двонаправленим списком в мові програмування C++ та його застосування
  + Статус: Ознайомлений
  + Початок опрацювання теми: 09.12.2023 16.00
  + Звершення опрацювання теми: 09.12.2023 17.40
* Тема №5: Бінарні дерева пошуку в C++.
  + Джерела Інформації:
    - Стаття: Introduction to Binary Tree [Introduction to Binary Tree - Data Structure and Algorithm Tutorials - GeeksforGeeks](https://www.geeksforgeeks.org/introduction-to-binary-tree-data-structure-and-algorithm-tutorials/)
    - Відео: C++ Теорія ⦁ Урок 144 ⦁ Бінарне дерево[C++ Теорія ⦁ Урок 144 ⦁ ADT ⦁ Бінарне дерево](https://www.youtube.com/watch?v=qBFzNW0ALxQ&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:
    - Опрацьовано та вивчено основні методи роботи з Бінарними деревами в мові програмування C++
    - Вивчено алгоритми обходу бінарних дерев
  + Статус: Ознайомлений
  + Початок опрацювання теми: 10.12.2023 14.50
  + Звершення опрацювання теми: 10.12.2023 17.10

# **Виконання роботи:**

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

Завдання №1 VNS Lab 10

* Варіант завдання: 11
* 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.

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

символів). Сформувати двонаправлений список. Знищити з нього елемент із

заданим ключем, додати елемент із зазначеним номером.

* Важливі деталі для врахування в імплементації програми:

Використання цілочисельних значень для запису в Список

Завдання №2 Algotester Lab 5

* Варіант завдання: 3
* Деталі завдання: <https://algotester.com/en/ContestProblem/DisplayWithEditor/134644>
* Важливі деталі для врахування в імплементації програми:

Дії над клітинками мапи використовувати двовимірний масив

Завдання №3 Algotester Lab 7-8

* Варіант завдання: 2
* Деталі завдання: <https://algotester.com/en/ContestProblem/DisplayWithEditor/134648>
* Важливі деталі для врахування в імплементації програми

Використання власноруч написаних функцій для обробки Динамічного Масиву

Завдання №4 Class Practice Work Task 1

* Варіант завдання: немає
* 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.

* Важливі деталі для врахування в імплементації програми

Завдання виконано на основі власноруч написаного Однозв’язного списку та власних функцій

Завдання №5 Class Practice Work Task 2

* Варіант завдання: немає
* Деталі завдання:
* 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

## -Задача №5 - Записати кожному батьківському вузлу суму підвузлів

void tree\_sum(TreeNode \*root);

*Умови задачі:*

- використовувати цілочисельні значення у вузлах дерева;

- реалізувати функцію, яка ітеративно проходить по бінарному дереві і записує у батьківський вузол суму значень підвузлів

- вузол-листок не змінює значення

- значення змінюються від листків до кореня дерева

* Важливі деталі для врахування в імплементації програми

Використання власноруч написаного Бінарного дерева та методів, які вимагалися в завданні

Завдання №6 Self Practice Work

* Варіант завдання: немає
* Деталі завдання: <https://algotester.com/en/ArchiveProblem/DisplayWithEditor/40269>
* Важливі деталі для врахування в імплементації програми

Використання типу даних string для запису даних та циклу для перевірки даних.

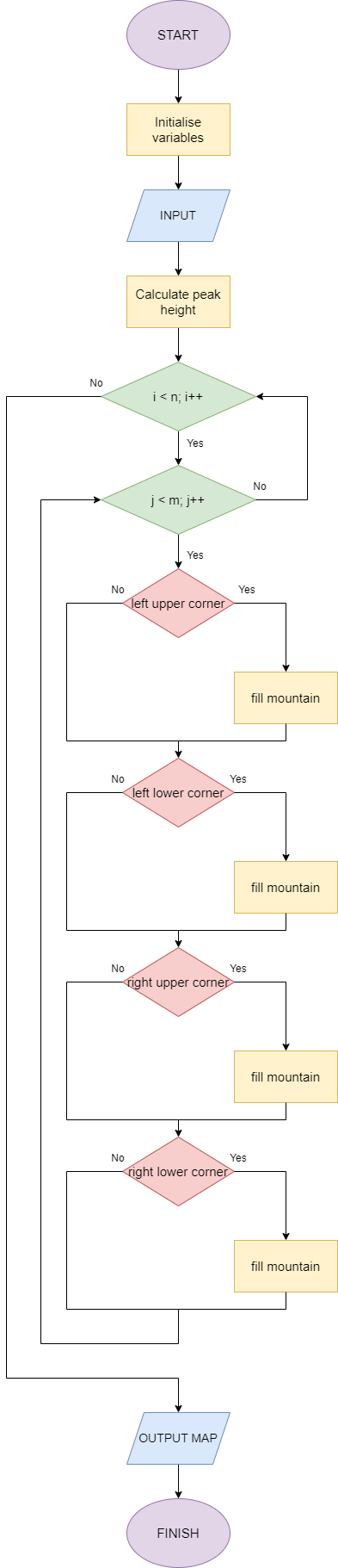
## **2. Дизайн та планована оцінка часу виконання завдань:**

Програма №1 VNS Lab 10

* Планований час на реалізацію: 120 хвилин

Програма №2 Algotester Lab 5

* Блок-схема



*Рисунок 1. Блок-схема до програми №2*

* Планований час на реалізацію: 100 хвилин

Програма №3 Algotester Lab 7-8

* Планований час на реалізацію: 120 хвилин

Програма №4 Class Practice Work Task 1

* Планований час на реалізацію: 200 хвилин

Програма №5 Class Practice Work Task 2

* Планований час на реалізацію: 200 хвилин

Програма №6 Self Practice Work

* Планований час на реалізацію: 40 хвилин

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

Для виконання поставлених завдань додаткова конфігурація середовища не є необхідною

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

Завдання №1 VNS Lab 10

#include <iostream>

#include <fstream>

#include <string.h>

using namespace std;

struct Node

{

const char\* data;

Node \*next;

Node \*prev;

Node(const char\* data) : data(data), next(nullptr), prev(nullptr)

{}

};

class DoublyLinkedList

{

public:

Node \*head;

Node \*tail;

DoublyLinkedList(): head(nullptr), tail(nullptr)

{}

void push\_back(const char\* value)

{

Node\* newNode = new Node(value);

if (tail == nullptr)

{

head = tail = newNode;

}

else

{

tail->next = newNode;

newNode->prev = tail;

tail = newNode;

}

}

void print(Node \*head)

{

Node \*current = head;

if (current == nullptr) {

cout << "List is empty" << endl;

return;

}

while (current != nullptr)

{

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

current = current->next;

}

cout << endl;

}

void delete\_element(const char\* key)

{

Node \*current = head;

while (current != nullptr)

{

Node\* nextNode = current->next;

if (strcmp(current->data, key) == 0)

{

if (current->prev != nullptr)

{

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

}

else

{

head = current->next;

if (head != nullptr)

{

head->prev = nullptr;

}

}

if (current->next != nullptr)

{

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

}

else

{

tail = current->prev;

if (tail != nullptr)

{

tail->next = nullptr;

}

}

delete current;

}

current = nextNode;

}

}

void add\_element(const char\* data, int number)

{

Node\* newNode = new Node(data);

if(number == 0)

{

newNode->next = head;

if(head != nullptr)

{

head->prev = newNode;

}

head = newNode;

if(tail == nullptr)

{

tail = newNode;

}

}

else

{

Node\* current = head;

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

{

current = current->next;

}

if (current != nullptr)

{

newNode->prev = current;

newNode->next = current->next;

if (current->next != nullptr)

{

current->next->prev = newNode;

}

else

{

tail = newNode;

}

current->next = newNode;

}

}

}

void write(const char\* file)

{

ofstream file\_1(file);

Node \*current = head;

while (current != nullptr)

{

file\_1.write(current->data, strlen(current->data) + 1);

//file\_1.put('\n');

current = current->next;

}

file\_1.close();

}

void read(const char\* file)

{

ifstream file\_1(file);

string line;

while (getline(file\_1, line))

{

push\_back(line.c\_str());

}

file\_1.close();

}

~DoublyLinkedList()

{

Node \*current = head;

Node \*next;

while (current != nullptr)

{

next = current->next;

delete current;

current = next;

}

head = tail = nullptr;

}

};

int main()

{

DoublyLinkedList linked\_list;

linked\_list.push\_back("hi");

linked\_list.push\_back("C++");

linked\_list.push\_back("programming");

linked\_list.push\_back("language");

cout << "Linked List:" << endl;

linked\_list.print(linked\_list.head);

linked\_list.delete\_element("programming");

cout << "Linked List after deleting:" << endl;

linked\_list.print(linked\_list.head);

linked\_list.add\_element("the best", 1);

cout << "Linked List after adding:" << endl;

linked\_list.print(linked\_list.head);

linked\_list.write("list.txt");

linked\_list.~DoublyLinkedList();

cout << "Linked List after destroying:" << endl;

linked\_list.print(linked\_list.head);

linked\_list.read("list.txt");

cout << "Linked List after restoring:" << endl;

linked\_list.print(linked\_list.head);

return 0;

}

Програма створює двозв’язний список та працює з ним у відповідності до поставленого завдання. В список додаються елементи, далі згідно з варіантом програма видаляє їх і додає нові. Список записується в файл, видаляється і відновлюється з файлу.

Посилання на файл програми у пул-запиті GitHub

https://github.com/artificial-intelligence-department/ai\_programming\_playground/pull/827

Завдання №2 Algotester Lab 5

#include <iostream>

#include <algorithm>

using namespace std;

int main()

{

int n, m, x, y;

cin >> n >> m;

cin >> x >> y;

int mountain[n][m];

int peak = max(x - 1, n - x) + max(y - 1, m - y);

x--;

y--;

mountain[x][y] = peak;

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

{

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

{

if (x - i >= 0 && x - i < n && y - j >= 0 && y - j < m)

{

mountain[x - i][y - j] = max(0, peak - (i + j));

}

if (x + i >= 0 && x + i < n && y + j >= 0 && y + j < m)

{

mountain[x + i][y + j] = max(0, peak - (i + j));

}

if (x - i >= 0 && x - i < n && y + j >= 0 && y + j < m)

{

mountain[x - i][y + j] = max(0, peak - (i + j));

}

if (x + i >= 0 && x + i < n && y - j >= 0 && y - j < m)

{

mountain[x + i][y - j] = max(0, peak - (i + j));

}

}

}

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

{

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

{

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

}

cout << endl;

}

return 0;

}

Програма отримує вхідні дані та ініціалізує двовимірний масив. Після цього за допомогою циклу і вкладеного циклу виконується розфарбування мапи. Після цього отримана мапа виводиться на екран.

Посилання на файл програми у пул-запиті GitHub

https://github.com/artificial-intelligence-department/ai\_programming\_playground/pull/827

Завдання №3 Algotester Lab 7-8

#include <iostream>

using namespace std;

class DynamicArray

{

private:

int size;

int capacity;

int \*arr;

public:

DynamicArray() : arr(nullptr), size(0), capacity(1)

{

arr = new int[capacity];

}

~DynamicArray()

{

}

void insert(int index, int num, int \*data)

{

if (size + num >= capacity)

{

while (size + num >= capacity)

{

capacity \*= 2;

}

int \*newArr = new int[capacity];

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

{

newArr[i] = arr[i];

}

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

{

newArr[index + i] = data[i];

}

for (int i = index; i < size; i++)

{

newArr[num + i] = arr[i];

}

delete[] arr;

arr = newArr;

}

else

{

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

{

arr[i + num] = arr[i];

}

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

{

arr[index + i] = data[i];

}

}

size = size + num;

}

void erase(int index, int num)

{

int left = size - index - num;

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

{

arr[index + i] = arr[index + i + num];

}

size -= num;

}

int get\_size()

{

return size;

}

int get\_capacity()

{

return capacity;

}

int& operator[](int index)

{

return arr[index];

}

int get(int index)

{

return arr[index];

}

void set(int index, int data)

{

arr[index] = data;

}

void print()

{

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

{

cout << arr[i] << " ";

}

cout << endl;

}

};

int main() {

int Q;

DynamicArray array;

cin >> Q;

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

{

string request;

cin >> request;

if (request == "insert")

{

int index, num;

cin >> index >> num;

int \*data = new int[num];

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

{

cin >> data[i];

}

array.insert(index, num, data);

delete[] data;

}

else if (request == "erase")

{

int index, num;

cin >> index >> num;

array.erase(index, num);

}

else if (request == "size")

{

cout << array.get\_size() << endl;

}

else if (request == "capacity")

{

cout << array.get\_capacity() << endl;

}

else if (request == "get")

{

int index;

cin >> index;

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

}

else if (request == "set")

{

int index, data;

cin >> index >> data;

array.set(index, data);

}

else if (request == "print")

{

array.print();

}

}

return 0;

}

Програма отримує вхідні дані згідно з шаблоном. Далі за допомогою розгалужень програма викликає відповідний метод і виводить результат роботи у консоль. Також видаляється виділене в пам’яті місце для тимчасових структур.

Посилання на файл програми у пул-запиті GitHub

https://github.com/artificial-intelligence-department/ai\_programming\_playground/pull/827

Завдання №4 Class Practice Work Task 1

#include <iostream>

using namespace std;

struct Node

{

Node \*next;

int data;

Node(int data) : data(data), next(nullptr)

{}

};

class LinkedList

{

private:

public:

Node \*head;

LinkedList() : head(nullptr)

{}

void push\_back(int data)

{

Node \*newNode = new Node(data);

if (head == nullptr)

{

head = newNode;

}

else

{

Node \*current = head;

while (current->next != nullptr)

{

current = current->next;

}

current->next = newNode;

}

}

void print(Node \*head)

{

Node \*current = head;

while (current != nullptr)

{

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

current = current->next;

}

cout << endl;

}

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;

}

return prev;

}

~LinkedList()

{

Node \*current = head;

Node \*next;

while (current != nullptr)

{

next = current->next;

delete current;

current = next;

}

head = 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;

}

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

{

Node \*result = nullptr;

int carry = 0;

while (n1 != nullptr || n2 != nullptr || carry != 0)

{

int sum;

if (n1)

{

sum = n1->data;

}

else

{

sum = 0;

}

if (n2)

{

sum += n2->data;

}

else

{

sum = 0;

}

sum += carry;

carry = sum / 10;

Node \*newNode = new Node(sum % 10);

newNode->next = result;

result = newNode;

if (n1)

{

n1 = n1->next;

}

if (n2)

{

n2 = n2->next;

}

}

return result;

}

int main()

{

LinkedList linked\_list;

LinkedList linked\_list\_2;

linked\_list.push\_back(9);

linked\_list.push\_back(7);

linked\_list.push\_back(3);

linked\_list.push\_back(4);

linked\_list.push\_back(5);

linked\_list\_2.push\_back(9);

linked\_list\_2.push\_back(7);

linked\_list\_2.push\_back(3);

linked\_list\_2.push\_back(4);

linked\_list\_2.push\_back(5);

LinkedList result\_linked\_list;

result\_linked\_list.head = add(linked\_list.head, linked\_list\_2.head);

result\_linked\_list.print(result\_linked\_list.head);

bool comparing\_result = compare(linked\_list.head, linked\_list\_2.head);

if (comparing\_result)

{

cout << "Linked lists are simmilar" << endl;

}

else

{

cout << "Linked lists are different" << endl;

}

linked\_list.print(linked\_list.head);

linked\_list.head = linked\_list.reverse(linked\_list.head);

linked\_list.print(linked\_list.head);

return 0;

}

У програмі описано власноруч створений однозв’язний список. Також реалізовано функції реверсу списку, порівняння та додавання великих чисел. Всі функції викликаються в main() і результат виконання виводиться в консоль

Посилання на файл програми у пул-запиті GitHub

https://github.com/artificial-intelligence-department/ai\_programming\_playground/pull/827

Завдання №5 Class Practice Work Task 2

#include <iostream>

using namespace std;

struct Node

{

int key;

Node \*left;

Node \*right;

};

void add\_tree\_node(Node \*&root, int key)

{

if (root == nullptr)

{

root = new Node{ key, nullptr, nullptr };

return;

}

if (key < root->key)

{

add\_tree\_node(root->left, key);

}

else

{

add\_tree\_node(root->right, key);

}

}

void in\_order(Node \*root)

{

if (root == nullptr)

{

return;

}

in\_order(root->left);

cout << root->key << " ";

in\_order(root->right);

}

Node \*create\_mirror\_flip(Node \*root)

{

if (root == nullptr)

{

return nullptr;

}

Node \*newRoot = new Node{ root->key, nullptr, nullptr };

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

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

return newRoot;

}

void tree\_sum(Node \*root)

{

if (root == nullptr || (root->left == nullptr && root->right == nullptr))

{

return;

}

if (root->left != nullptr || root->right != nullptr)

{

int sum = 0;

if (root->left != nullptr)

{

sum += root->left->key;

}

if (root->right != nullptr)

{

sum += root->right->key;

}

root->key = sum;

}

tree\_sum(root->left);

tree\_sum(root->right);

}

int main()

{

Node \*root = nullptr;

add\_tree\_node(root, 5);

add\_tree\_node(root, 2);

add\_tree\_node(root, 3);

add\_tree\_node(root, 4);

add\_tree\_node(root, 1);

in\_order(root);

cout << endl;

Node \*mirror\_Root = create\_mirror\_flip(root);

in\_order(mirror\_Root);

cout << endl;

tree\_sum(root);

in\_order(root);

return 0;

}

У програмі реалізовано бінарне дерево пошуку а також методи для нього(віддзеркалення та обрахунок суми чайлдів для парента). Також реалізовано метод для інфіксного обходу дерева для можливості виводу в консоль. Основна функція додає вузли в дерево та викликає вищезгадані методи, виводячи результат в консоль

Посилання на файл програми у пул-запиті GitHub

https://github.com/artificial-intelligence-department/ai\_programming\_playground/pull/827

Завдання №6 Self Practice Work

#include <iostream>

#include <string>

int main()

{

std::string text;

int res = 0;

std::cin >> text;

for(int i = 0; i < text.length(); i++)

{

if(text[i] != text[i - 1])

{

res++;

}

}

std::cout << res;

return 0;

}

Програма отримує з консолі рядок. Далі за допомогою циклу знаходяться унікальні символи в рядку. Результат виконання виводиться в консоль

Посилання на файл програми у пул-запиті GitHub

https://github.com/artificial-intelligence-department/ai\_programming\_playground/pull/827

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

Завдання №1 VNS Lab 10

INPUT/OUTPUT:

Linked List:

hi C++ programming language

Linked List after deleting:

hi C++ language

Linked List after adding:

hi C++ the best language

Linked List after destroying:

List is empty

Linked List after restoring:

hi C++ the best language

Час затрачений на виконання завдання: 150 хвилин

Завдання №2 Algotester Lab 5

INPUT:

3 9

1 2

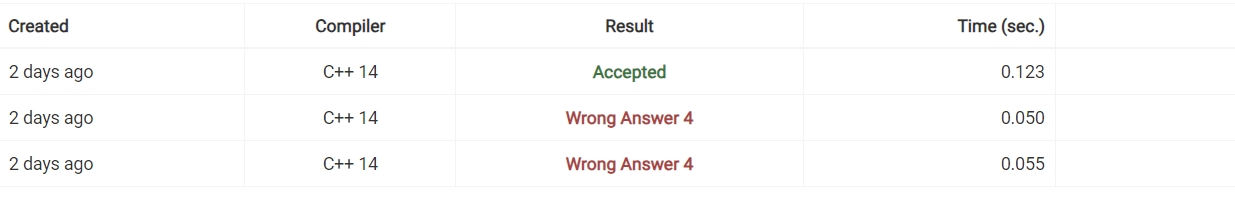
OUTPUT:

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

Час затрачений на виконання завдання: 90 хвилин



*Рисунок 2. Результат зарахування програми №2 на Алготестері*

Завдання №3 Algotester Lab 7-8

INPUT:

12

size

capacity

insert 0 2

100 100

size

capacity

insert 0 2

102 102

size

capacity

insert 0 2

103 103

size

capacity

print

OUTPUT:

0

1

2

4

4

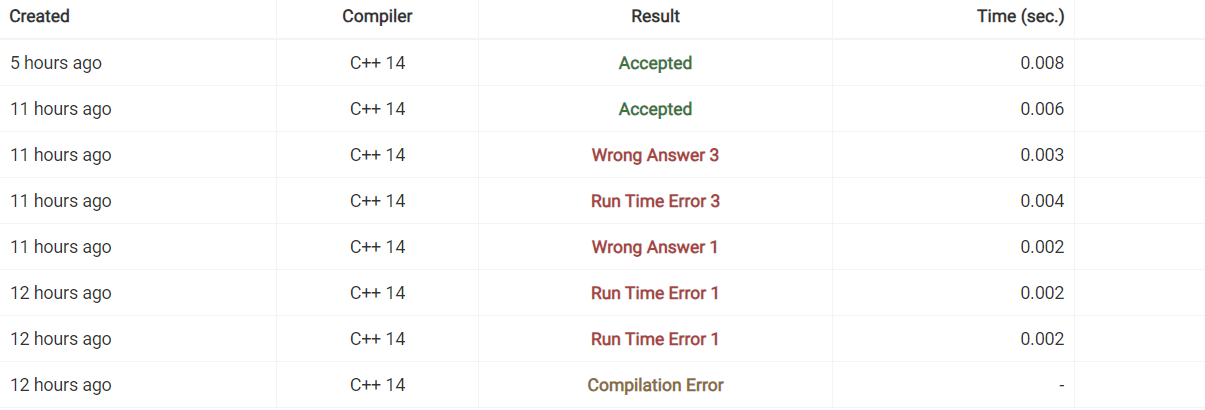
8

6

8

103 103 102 102 100 100

Час затрачений на виконання завдання: 200 хвилин



*Рисунок 3. Результат зарахування програми №3 на Алготестері*

Завдання №4 Class Practice Work Task 1

INPUT/OUTPUT:

1 0 8 7 5 8

Linked lists are simmilar

9 7 3 4 5

5 4 3 7 9

Час затрачений на виконання завдання: 120 хвилин

Завдання №5 Class Practice Work Task 2

INPUT/OUTPUT:

1 2 3 4 5

5 4 3 2 1

1 4 4 4 2

Час затрачений на виконання завдання: 110 хвилин

Завдання №6 Self Practice Work

INPUT:

aabbbc

OUTPUT:

3

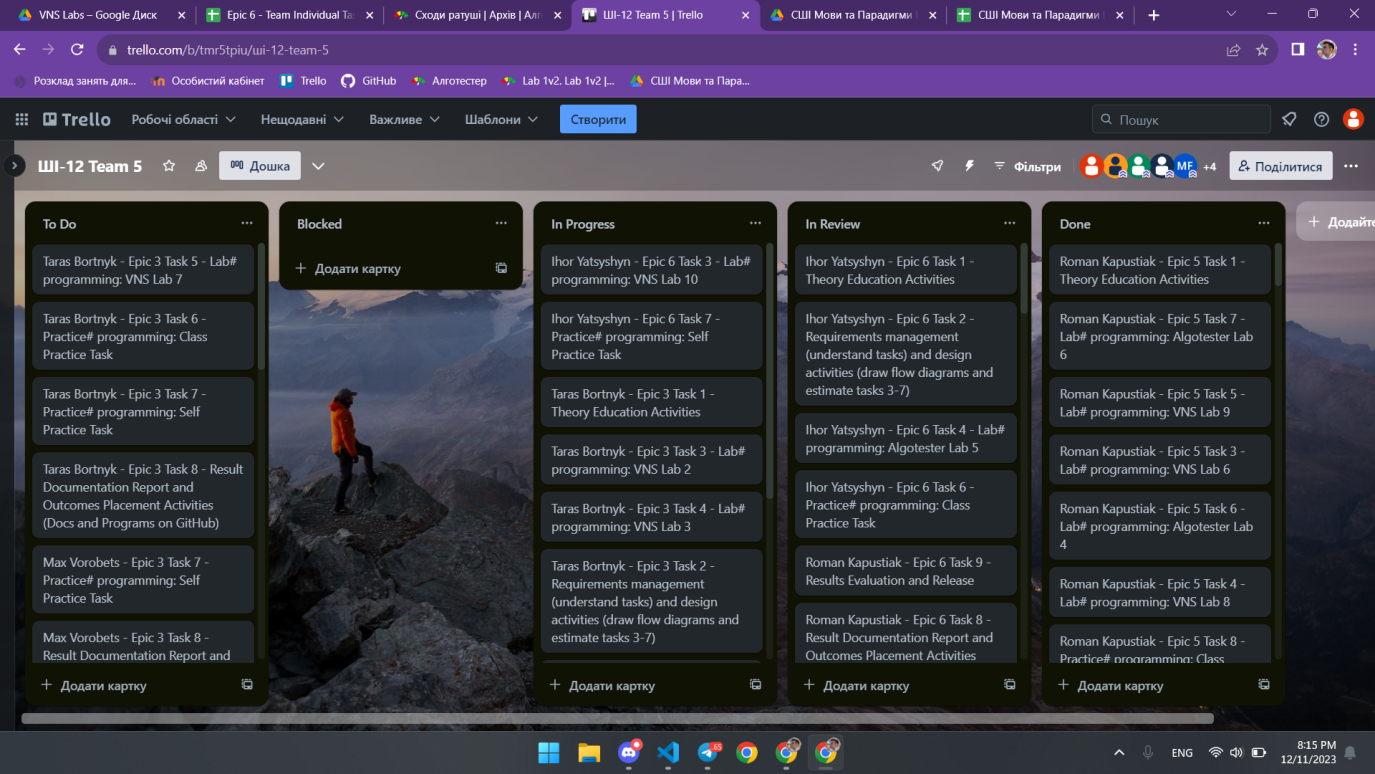
Час затрачений на виконання завдання: 25 хвилин

## **6. Кооперація з командою:**

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

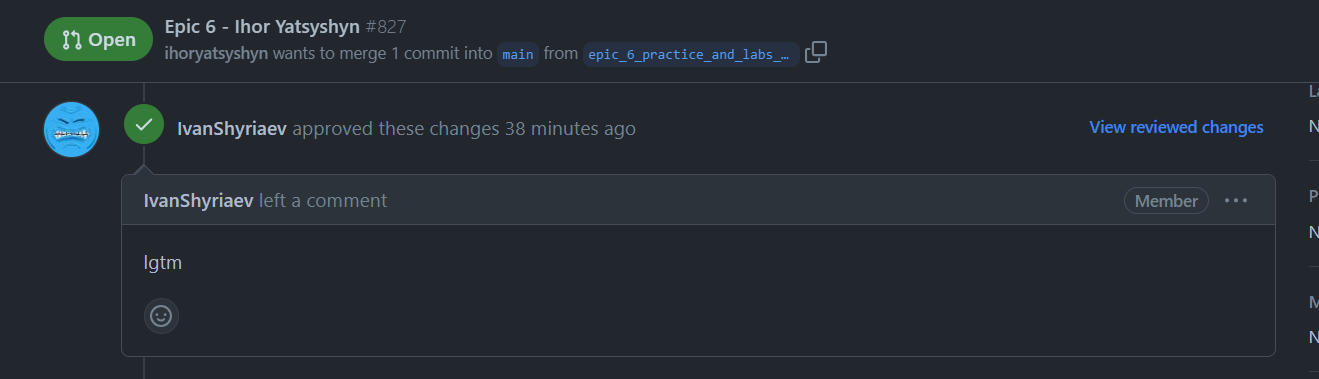


*Рисунок 4. Скріншот першої зустрічі*

**

*Рисунок 5. Скріншот Trello на час першої зустрічі*

* Скрін з коментарями від учасників команди на пул-реквесті з Рев’ю Роботи



*Рисунок 6. Скріншот коментарів від учасників команди до пул-реквесту*

# **Висновки:**

Під час опрацювання теоретичного матеріалу та роботи над завданнями розділу я навчився працювати з динамічними структурами даних в мові C++. Під час опрацювання матеріалу суттєвих труднощів не виникло. Весь пройдений матеріал закріплено практично завдяки виконанні лабораторних та практичних робіт а також самопрактиці.