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:***

Cтудент групи ШІ-14

Грицишин Максим Іванович

# **Тема роботи:**

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

# **Мета роботи:**

* Theory Education Activities
* Requirements management (understand tasks) and design activities (draw flow diagrams and estimate tasks 3-7)
* Lab# programming: VNS Lab 10
* Lab# programming: Algotester Lab 5
* Lab# programming: Algotester Lab 7-8
* Practice# programming: Class Practice Task
* Practice# programming:  Self Practice Task
* Result Documentation Report and Outcomes Placement Activities (Docs and Programs on GitHub)
* Results Evaluation and Release

# **Теоретичні відомості:**

1. Теоретичні відомості з переліком важливих тем:

* Тема №1: Списки.
* Тема №2: Дерева
* Тема №3: Стек
* Тема №4: Черга

1. Індивідуальний план опрацювання теорії:

* Тема №1: Списки.
  + Джерела Інформації
    - <https://www.youtube.com/watch?v=HKfj0l7ndbc&list=PL43pGnjiVwgS5njI0HMGnqSH18tSSuLz_&index=3&pp=iAQB>
    - <https://www.youtube.com/watch?v=RNMIDj62o_o&list=PL43pGnjiVwgS5njI0HMGnqSH18tSSuLz_&index=4&pp=iAQB>
    - <https://www.youtube.com/watch?v=-StYr9wILqo&list=PL43pGnjiVwgS5njI0HMGnqSH18tSSuLz_&index=8&pp=iAQB>
    - ChatGPT
  + Що опрацьовано:
    - Ознайомлено з теорією
    - Використано у практичних та лабораторних роботах
  + Статус: Ознайомлений
  + Початок опрацювання теми: 12.12.23
  + Звершення опрацювання теми: 13.12.23
* Тема №2: Дерева
  + Джерела Інформації:
    - <https://www.youtube.com/watch?v=UHxtjVsOTHc>
    - <https://youtu.be/_IhTp8q0Mm0?si=Ry5wReFC1KeM3icl>
    - ChatGPT
  + Що опрацьовано:
    - Ознайомлено з теорією
    - Використано у практичних та лабораторних роботах
  + Статус: Ознайомлений
  + Початок опрацювання теми: 12.12.23
  + Завершення опрацювання теми: 13.12.23
* Тема №3: Стек
  + Джерела Інформації:
    - [https://www.youtube.com/watch?v=GBST5uQ\_yos&list=PL43pGnjiVwgS5njI0HMGnqSH18tSSuLz\_&index=5&pp=iAQB](https://www.youtube.com/watch?v=GBST5uQ_yos&list=PL43pGnjiVwgS5njI0HMGnqSH18tSSuLz_&index=5&pp=iAQB%20)
    - ChatGPT
    - Інтернет-ресурси
  + Що опрацьовано:
    - Ознайомлено з теорією
  + Статус: Ознайомлений
  + Початок опрацювання теми: 12.12.23
  + Завершення опрацювання теми: ---
* Тема №4 Черга
  + Джерела Інформації:
    - <https://www.youtube.com/watch?v=jaK4pn1jXTo&list=PL43pGnjiVwgS5njI0HMGnqSH18tSSuLz_&index=6&pp=iAQB>
    - ChatGPT
    - Інтернет-ресурси
  + Що опрацьовано:
    - Ознайомлений з теорією
  + Статус: Ознайомлений
  + Початок опрацювання теми: 13.12.23
  + Завершення опрацювання теми: ---

# **Виконання роботи:**

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

Завдання №1 Лабораторна робота №10 "Інформаційні динамічні структури"

* Варіант завдання: 6
* Деталі завдання: Записи в лінійному списку містять ключове поле типу int. Сформувати двонаправлений список. Знищити з нього елемент із заданим номером, додати елемент у початок списку.
* Важливі деталі для врахування в імплементації програми: Написати програму, у якій створюються динамічні структури й виконати їхню обробку.

Потрібно розробити такі функції:

1. Створення списку.

2. Додавання елемента в список (у відповідності зі своїм варіантом).

3. Знищення елемента зі списку (у відповідності зі своїм варіантом).

4. Друк списку.

5. Запис списку у файл.

6. Знищення списку.

7. Відновлення списку з файлу.

Завдання №2 Лабораторна робота №5 у Algotester

* Варіант завдання: 3
* Деталі завдання: Дано карту гори розмiром N × M. Також відомі координати {x, y} , у яких знаходиться вершина гори. Завдання - розмалювати карту таким чином, щоб найнижча точка мала число 0, а пiк гори мав найбiльше число. Клiтинки, якi мають сумiжну сторону з вершиною мають висоту на один меншу, сумiжнi з ними i не розфарбованi мають ще на 1 меншу висоту i так далi.
* Важливі деталі для врахування в імплементації програми:

Вхiднi данi:

У першому рядку 2 числа N та M - розмiри карти

у другому рядку 2 числа x та y - координати пiку гори

Вихiднi данi:

N рядкiв по M елементiв в рядку через пробiл - висоти карти.

Завдання №3 Лабораторна робота №7\_8 у Algotester

* Варіант завдання: 1
* Деталі завдання: Ваше завдання - власноруч реалiзувати структуру даних "Двозв’язний список". Ви отримаєте Q запитiв, кожен запит буде починатися зi слова-iдентифiкатора, пiсля якого йдуть його аргументи.

Вам будуть поступати запити такого типу:

• Вставка:

Iдентифiкатор - insert

Ви отримуєте цiле число index елемента, на мiсце якого робити вставку.

Пiсля цього в наступному рядку рядку написане число N - розмiр списку, який треба вста-

Вити. У третьому рядку N цiлих чисел - список, який треба вставити на позицiю index.

• Видалення:

Iдентифiкатор - erase

Ви отримуєте 2 цiлих числа - index, iндекс елемента, з якого почати видалення та n -

кiлькiсть елементiв, яку треба видалити.

• Визначення розмiру:

Iдентифiкатор - size

Ви не отримуєте аргументiв. Ви виводите кiлькiсть елементiв у списку.

• Отримання значення i-го елементу

Iдентифiкатор - get

Ви отримуєте цiле число - index, iндекс елемента. Ви виводите значення елемента за iндексом.

• Модифiкацiя значення i-го елементу

Iдентифiкатор – set.

Ви отримуєте 2 цiлих числа - iндекс елемента, який треба змiнити, та його нове значення.

• Вивiд списку на екран

Iдентифiкатор - print

Ви не отримуєте аргументiв. Ви виводите усi елементи списку через пробiл. Реалiзувати використовуючи перегрузку оператора <<

* Важливі деталі для врахування в імплементації програми:

Вхiднi данi:

Цiле число Q - кiлькiсть запитiв. У наступних рядках Q запитiв у зазначеному в умовi форматi.

Вихiднi данi:

Вiдповiдi на запити у зазначеному в умовi форматi.

Важлива примітка:

Гарантується, що усi данi коректнi. Виходу за межi списку або розмiр, бiльший нiж розмiр списку недопустимi. Iндекси починаються з нуля.

Завдання №4 Практична робота №1 "Зв’язаний список: Реверс списку"

* Деталі завдання: ***Реалізувати метод реверсу списку:*** Node\* reverse(Node \*head);

*Умови задачі:*

1. використовувати цілочисельні значення в списку;
2. реалізувати метод реверсу;
3. реалізувати допоміжний метод виведення вхідного і обернутого списків;

* Важливі деталі для врахування в імплементації програми: ---

Завдання №5 Практична робота №2 “Зв'язаний список: Порівняння списків”

* Деталі завдання: bool compare(Node \*h1, Node \*h2);

*Умови задачі:*

1. використовувати цілочисельні значення в списку;
2. реалізувати функцію, яка ітеративно проходиться по обох списках і порівнює дані в кожному вузлі;
3. якщо виявлено невідповідність даних або якщо довжина списків різна (один список закінчується раніше іншого), функція повертає ***false***.

* Важливі деталі для врахування в імплементації програми: ---

Завдання №6 Практична робота №3 “Зв'язаний список: Додавання великих чисел”

* Деталі завдання: Node\* add(Node \*n1, Node \*n2);

*Умови задачі:*

1. використовувати цифри від 0 до 9 для значень у списку;
2. реалізувати функцію, яка обчислює суму двох чисел, які збережено в списку; молодший розряд числа записано в голові списка (напр. 379  ⟹  9→7→3);
3. функція повертає новий список, передані в функцію списки не модифікуються.

* Важливі деталі для врахування в імплементації програми: ---

Завдання №7 Практична робота №4 “Бінарні дерева: Віддзеркалення дерева”

* Деталі завдання: TreeNode \*create\_mirror\_flip(TreeNode \*root);

*Умови задачі:*

1. використовувати цілі числа для значень у вузлах дерева
2. реалізувати функцію, що проходить по всіх вузлах дерева і міняє місцями праву і ліву вітки дерева
3. функція повертає нове дерево, передане в функцію дерево не модифікується

* Важливі деталі для врахування в імплементації програми: ---

Завдання №8 Практична робота №5 “Бінарні дерева: Записати кожному батьківському вузлу суму підвузлів”

* Деталі завдання: void tree\_sum(TreeNode \*root);

*Умови задачі:*

1. використовувати цілочисельні значення у вузлах дерева;
2. реалізувати функцію, яка ітеративно проходить по бінарному дереві і записує у батьківський вузол суму значень підвузлів
3. вузол-листок не змінює значення
4. значення змінюються від листків до кореня дерева

* Важливі деталі для врахування в імплементації програми: ---

Завдання №9 Власна практична робота «Зуби»

Деталі завдання: Мале Бісеня любить гострити зуби. А Зла Тітонька любить до нього підходити і питатися: «Що, зуби гостриш?». Бісеняті таке не дуже подобається, тому воно придумало робити таке.

У Малого Бісеняти є n зубів. Кожен зуб має коефіцієнт загостреності ai. Також існує межа загостреності k. Якщо коефіцієнт загостреності певного зуба є більшим чи рівним межі загостреності, то такий зуб вважається загостреним.

Мале Бісеня хоче наступного разу, коли Зла Тітонька його щось запитає, показати їй якнайбільше загострених зубів, що розташовані поспіль. Допоможіть Малому Бісеняті дізнатися, скільки найбільше зубів воно зможе показати.

* Важливі деталі для врахування в імплементації програми:

# Вхідні дані:

У першому рядку задані два цілих числа n та k — кількість зубів та межа загостреності відповідно.

В другому рядку задано n цілих чисел ai — коефіцієнти загостреності зубів.

# Вихідні дані:

Єдине ціле число — відповідь на задачу.

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

Програма №1 Лабораторна робота №10 "Інформаційні динамічні структури"

* Запланований час на реалізацію: 4год
* Важливі деталі для врахування в імплементації:

1. Написати функцію для друку списку. Функція повинна передбачати вивід

повідомлення, якщо список порожній.

2. Написати функції для знищення й додавання елементів списку у

відповідності зі своїм варіантом

3. Виконати зміни в списку й друк списку після кожної зміни.

4. Написати функцію для запису списку у файл.

5. Написати функцію для знищення списку.

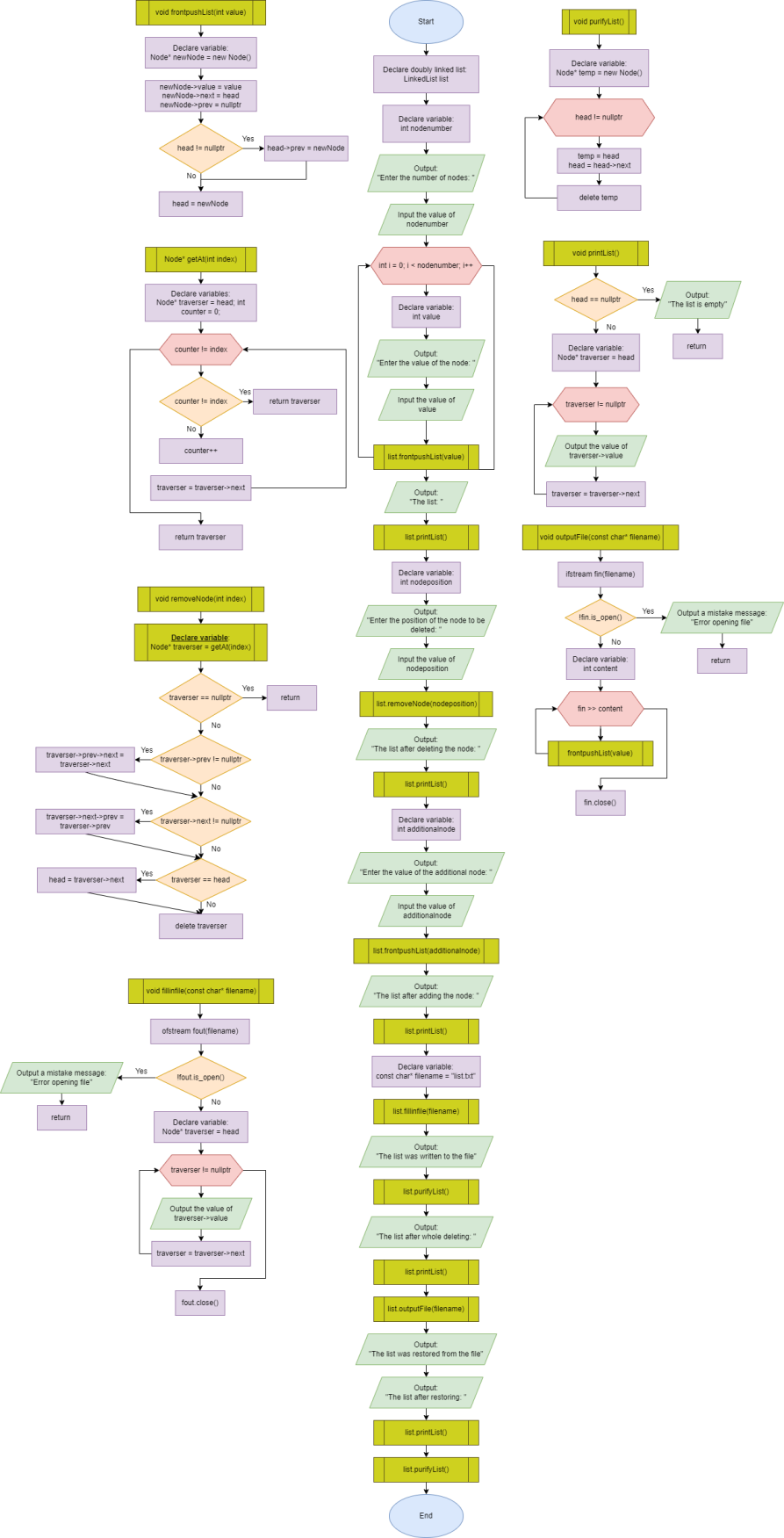
6. Записати список у файл, знищити його й виконати друк (при друці повинне

бути видане повідомлення "Список порожній").

7. Написати функцію для відновлення списку з файлу.

8. Відновити список і роздрукувати його.

9.Знищити список.



Малюнок : Блок-схема до програми №1

Програма №2 Лабораторна робота №5 у Algotester

* Планований час на реалізацію: 3год
* Важливі деталі для врахування в імплементації: обмеження у пам’яті, часі виконання та у значеннях змінних

Програма №3 Лабораторна робота №7\_8 у Algotester

* Планований час на реалізацію: 5год
* Важливі деталі для врахування в імплементації: Використовувати STL заборонено.

Програма №4 Практична робота №1 "Зв’язаний список: Реверс списку"

* Планований час на реалізацію: 2год
* Важливі деталі для врахування в імплементації: Спочатку ми визначаємо просту структуру ***Node*** для нашого пов’язаного списку. Потім функція ***reverse*** ітеративно змінює список, маніпулюючи наступними покажчиками кожного вузла. ***printList*** — допоміжна функція для відображення списку. Основна функція створює зразок списку, демонструє реверсування та друкує вихідний і обернений списки.

Програма №5 Практична робота №2 “Зв'язаний список: Порівняння списків”

* Планований час на реалізацію: 2год
* Важливі деталі для врахування в імплементації:
* Для пов’язаного списку визначено структуру ***Node***.
* Функція ***compare*** ітеративно проходить обидва списки одночасно, порівнюючи дані в кожному вузлі.
* Якщо виявлено невідповідність даних або якщо довжина списків різна (один список закінчується раніше іншого), функція повертає ***false***.
* Основна функція ***main*** створює два списки та демонструє порівняння.

Програма №6 Практична робота №3 “Зв'язаний список: Додавання великих чисел”

* Планований час на реалізацію: 2год
* Важливі деталі для врахування в імплементації: ---

Програма №7 Практична робота №4 “Бінарні дерева: Віддзеркалення дерева”

* Планований час на реалізацію: 2год
* Важливі деталі для врахування в імплементації: ---

Програма №8 Практична робота №5 “Бінарні дерева: Записати кожному батьківському вузлу суму підвузлів”

* Планований час на реалізацію: 2год
* Важливі деталі для врахування в імплементації: ---

Програма №9 Власна практична робота “Зуби”

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

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

*Конфігурація з Епіку 1 не змінилась.*

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

Завдання №1 Лабораторна робота №10 "Інформаційні динамічні структури"

#include <iostream>

#include <fstream>

using namespace std;

class Node {

public:

    int value;

    Node\* prev, \* next;

};

class LinkedList{

private:

    Node\* head;

public:

    LinkedList() {

        head = nullptr;

    }

    void frontpushList(int value) {

        Node\* newNode = new Node();

        newNode->value = value;

        newNode->next = head;

        newNode->prev = nullptr;

        if (head != nullptr)

            head->prev = newNode;

        head = newNode;

    }

    Node\* getAt(int index) {

        Node\* traverser = head;

        int counter = 0;

        while (counter != index) {

            if (traverser == nullptr) return traverser;

            counter++;

            traverser = traverser->next;

        }

        return traverser;

    }

    void removeNode(int index) {

        Node\* traverser = getAt(index);

        if (traverser == nullptr) return;

        if (traverser->prev != nullptr) traverser->prev->next = traverser->next;

        if (traverser->next != nullptr) traverser->next->prev = traverser->prev;

        if (traverser == head) head = traverser->next;

        delete traverser;

    }

    void fillinfile(const char\* filename) {

        ofstream fout(filename);

        if (!fout.is\_open()) {

            cerr << "Error opening file" << endl;

            return;

        }

        Node\* traverser = head;

        while (traverser != nullptr) {

            fout << traverser->value << " ";

            traverser = traverser->next;

        }

        fout.close();

    }

    void purifyList() {

        Node\* temp = new Node();

        while (head != nullptr) {

            temp = head;

            head = head->next;

            delete temp;

        }

    }

    void printList() {

        if (head == nullptr) {

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

            return;

        }

        Node\* traverser = head;

        while (traverser != nullptr) {

            cout << traverser->value << " ";

            traverser = traverser->next;

        }

    }

    void outputFile(const char\* filename) {

        ifstream fin(filename);

        if (!fin.is\_open()) {

            cerr << "Error opening file" << endl;

            return;

        }

        int content;

        while (fin >> content) {

            frontpushList(content);

        }

        fin.close();

    }

};

int main() {

   LinkedList list;

   int nodenumber;

    cout << "Enter the number of nodes: ";

    cin >> nodenumber;

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

        int value;

        cout << "Enter the value of the node: ";

        cin >> value;

        list.frontpushList(value);

    }

    cout << "The list: ";

    list.printList();

    cout << endl;

    int nodeposition;

    cout << "Enter the position of the node to be deleted: ";

    cin >> nodeposition;

    list.removeNode(nodeposition);

    cout << "The list after deleting the node: ";

    list.printList();

    cout << endl;

    int additionalnode;

    cout << "Enter the value of the additional node: ";

    cin >> additionalnode;

    list.frontpushList(additionalnode);

    cout << "The list after adding the node: ";

    list.printList();

    cout << endl;

    const char\* filename = "list.txt";

    list.fillinfile(filename);

    cout << "The list was written to the file" << endl;

    list.purifyList();

    cout << "The list after whole deleting: ";

    list.printList();

    cout << endl;

    list.outputFile(filename);

    cout << "The list was restored from the file" << endl;

    cout << "The list after restoring: ";

    list.printList();

    cout << endl;

    list.purifyList();

    return 0;

}

Покликання на пул-реквест: <https://github.com/artificial-intelligence-department/ai_programming_playground/pull/956/files#diff-e4fa8f42678cc12d899b7957ba2c827fbb49e8c885464e71bbf182b1d80bbb39>

Завдання №2 Лабораторна робота №5 у Algotester

#include <iostream>

#include <vector>

#include <algorithm>

using namespace std;

int calculate(int i, int j, int x, int y) {

    return abs(i - x) + abs(j - y);

}

int main() {

    int N, M, x, y;

    std::cin >> N >> M >> x >> y;

    --x;

    --y;

    vector<vector<int>> map(N, vector<int>(M, 0));

    for (auto& row : map) {

        for (int& val : row) {

            val = calculate(&row - &map[0], &val - &row[0], x, y);

        }

    }

    int peak = 0;

    for (const auto& row : map) {

        for (const int& val : row) {

            peak = max(peak, val);

        }

    }

    for (const auto& row : map) {

        for (const int& val : row) {

            cout << peak - val << ' ';

        }

        cout << endl;

    }

    return 0;

}

Покликання на пул-реквест: <https://github.com/artificial-intelligence-department/ai_programming_playground/pull/956/files#diff-fcd5f83499762677817715855a37742f0a2c8de1fd63b193bb35df02edf6e3f2>

Завдання №3 Лабораторна робота №7\_8 у Algotester

#include  <iostream>

using namespace std;

class Node {

public:

     int value;

     Node\* prev, \* next;

};

class LinkedList {

private:

    struct Node{

        int value;

        Node\* next;

        Node\* prev;

        Node(int val, Node \*node = nullptr ) {

            value = val;

            next = node;

        }

    };

    Node\* head, \* tail;

    int size;

public:

    LinkedList() {

        head = nullptr;

        tail = nullptr;

        size = 0;

    }

     Node\* getNode(int index) {

        Node\* traverser = head;

        int counter = 0;

        while (counter != index) {

            if (traverser == nullptr) return traverser;

            counter++;

            traverser = traverser->next;

        }

        return traverser;

    }

    int getNodes(int index) {

        Node\* traverser = head;

        int counter = 0;

        while (counter != index) {

            if (traverser == nullptr) return traverser->value;

            counter++;

            traverser = traverser->next;

        }

        return traverser->value;

    }

    int listsize() {

        int size = 0;

        Node\* traverser = head;

        while (traverser != nullptr) {

            size++;

            traverser = traverser->next;

        }

        return size;

    }

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

        if (index > listsize()) {

            return;

        }

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

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

            traverser->value = arr[i];

            traverser->prev = nullptr;

            traverser->next = nullptr;

            if (index == 0) {

                traverser->next = head;

                if (head != nullptr) {

                    head->prev = traverser;

                }

                head = traverser;

                if (tail == nullptr) {

                    tail = traverser;

                }

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

                traverser->prev = tail;

                if (tail != nullptr) {

                    tail->next = traverser;

                }

                tail = traverser;

                if (head == nullptr) {

                    head = traverser;

                }

            } else {

                Node\* ptr = head;

                for (int j = 0; j < index - 1; j++) {

                    ptr = ptr->next;

                }

                traverser->next = ptr->next;

                traverser->prev = ptr;

                if (ptr->next != nullptr) {

                    ptr->next->prev = traverser;

                }

                ptr->next = traverser;

                }

            size = listsize() + 1;

            }

        }

    void eraseNode(int index, int N) {

        if (index < 0 || N < 0) return;

        Node\* traverser = getNode(index);

        for (int i = 0; i < N && traverser != nullptr; i++) {

        Node\* nextNode = traverser->next;

        Node\* prevNode = traverser->prev;

        if (prevNode)

            prevNode->next = nextNode;

        else

            head = nextNode;

        if (nextNode)

            nextNode->prev = prevNode;

        else

            tail = prevNode;

        delete traverser;

        traverser = nextNode;

    }

    }

    void setNode(int index, int content) {

        Node\* traverser = getNode(index);

        if (traverser != nullptr)

        traverser->value = content;

    }

    friend ostream& operator<<(ostream& os, const LinkedList& list) {

    Node \*ptr = list.head;

    while (ptr != nullptr) {

      os << ptr->value << " ";

      ptr = ptr->next;

    }

    return os;

  }

};

int main() {

    int Q;

    cin >> Q;

    LinkedList list;

    while (Q--) {

        string request;

        cin >> request;

        if(request == "insert") {

            int index;

            cin >> index;

            int N;

            cin >> N;

            int nums[N];

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

                cin >> nums[i];

            }

            list.insert(index, N, nums);

        }

        else if(request == "erase") {

            int index;

            int n;

            cin >> index >> n;

            cout << endl;

            list.eraseNode(index, n);

        }

        else if(request == "size") {

            int size = list.listsize();

            cout << size << endl;

        }

        else if(request == "set") {

            int index;

            int content;

            cin >> index >> content;

            list.setNode(index, content);

        }

        else if(request == "get") {

            int index;

            cin >> index;

             cout << list.getNodes(index) << endl;

        }

        else if(request == "print") {

            cout << list << endl;

            cout << endl;

        }

    }

    return 0;

}

Покликання на пул-реквест: <https://github.com/artificial-intelligence-department/ai_programming_playground/pull/956/files#diff-fb0ce8f53f799fe9f7228b9ced56ace785f058e4cdc93d13249d73bc7ecd8e48>

Завдання №4 Практична робота №1 "Зв’язаний список: Реверс списку"

#include <iostream>

using namespace std;

class Node {

public:

    int value;

    Node\* next;

};

Node\* reverse(Node \*head) {

    Node \*current = head;

    Node \*prev = nullptr, \*next = nullptr;

    while (current != nullptr) {

        next = current->next;

        current->next = prev;

        prev = current;

        current = next;

    }

    head = prev;

    return head;

}

    void printList(Node \*n) {

        if (n == nullptr) {

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

            return;

        }

        while (n != nullptr) {

            cout << n->value << " ";

            n = n->next;

        }

    }

int main() {

    Node\* head = new Node;

    Node\* second = new Node();

    Node\* third = new Node();

    head->value = 1;

    head->next = second;

    second->value = 2;

    second->next = third;

    third->value = 3;

    third->next = nullptr;

    cout << "Original list: ";

    printList(head);

    cout << endl;

    cout << "Reversed list: ";

    head = reverse(head);

    printList(head);

    cout << endl;

    return 0;

}

Покликання на пул-реквест: <https://github.com/artificial-intelligence-department/ai_programming_playground/pull/956/files#diff-305820193bc7e286d6001a955bfd564822e960065a315fc8ec48a921fccb4327>

Завдання №5 Практична робота №2 “Зв'язаний список: Порівняння списків”

#include <iostream>

using namespace std;

class Node {

public:

    int value;

    Node \* next;

    Node() : value(0), next(nullptr) {}

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

};

bool compare(Node \*a, Node \*b) {

        while (a != nullptr && b != nullptr) {

            if (a->value != b->value) {

                return false;

            }

            a = a->next;

            b = b->next;

        }

        return (a == nullptr && b == nullptr);

}

int main() {

    Node \*head1 = new Node(2);

    head1->next = new Node(4);

    head1->next->next = new Node(6);

    Node \*head2 = new Node(2);

    head2->next = new Node(4);

    head2->next->next = new Node(7);

    if(compare(head1, head2)) {

        cout << "Lists are equal" << endl;

    } else {

        cout << "Lists are not equal" << endl;

    }

    return 0;

}

Покликання на пул-реквест: <https://github.com/artificial-intelligence-department/ai_programming_playground/pull/956/files#diff-599bc2b54923fadfbd9b168142c80e6bcdbcea408bbe1eed5a1fadc7794cb680>

Завдання №6 Практична робота №3 “Зв'язаний список: Додавання великих чисел”

#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\* resultHead = nullptr;

    Node\* resultTail = nullptr;

    int carry = 0;

    while (n1 || n2 || carry) {

        int value1;

        if (n1) {

            value1 = n1->data;

        } else {

            value1 = 0;

        }

        int value2;

        if (n2) {

            value2 = n2->data;

        } else {

            value2 = 0;

        }

        int sum = value1 + value2 + carry;

        int digit = sum % 10;

        carry = sum / 10;

        if (n1 != nullptr) {

            n1 = n1->next;

        }

        if (n2 != nullptr) {

            n2 = n2->next;

        }

        if (resultHead == nullptr) {

            resultHead = new Node(digit);

            resultTail = resultHead;

        } else {

            resultTail->next = new Node(digit);

            resultTail = resultTail->next;

        }

    }

    return resultHead;

}

Node\* reverse(Node \*head) {

    Node \*current = head;

    Node \*prev = nullptr, \*next = nullptr;

    while (current != nullptr) {

        next = current->next;

        current->next = prev;

        prev = current;

        current = next;

    }

    head = prev;

    return head;

}

void printList(Node \*n) {

        if (n == nullptr) {

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

            return;

        }

        while (n != nullptr) {

            cout << n->data << " ";

            n = n->next;

        }

    }

int main() {

    Node\* number1 = new Node(3);

    number1->next = new Node(2);

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

    Node\* number2 = new Node(7);

    number2->next = new Node(8);

    number2->next->next = new Node(9);

    cout << "Number 1: ";

    printList(number1);

    cout << endl;

    cout << "Number 2: ";

    printList(number2);

    cout << endl;

    Node\* result1 = add(number1, number2);

    result1 = reverse(result1);

    cout << "Sum: ";

    printList(result1);

    cout << endl;

    return 0;

}

Покликання на пул-реквест: <https://github.com/artificial-intelligence-department/ai_programming_playground/pull/956/files#diff-4519c45919d7e916e19011c3bf83822034e306ee1fbd092c424f5363bcd8e25a>

Завдання №7 Практична робота №4 “Бінарні дерева: Віддзеркалення дерева”

#include <iostream>

using namespace std;

struct TreeNode {

    int data;

    TreeNode\* left;

    TreeNode\* right;

};

TreeNode\* createNode(int data) {

    TreeNode\* newNode = new TreeNode();

    newNode->data = data ;

    newNode->left = newNode->right = nullptr;

    return newNode;

}

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

    if (root == nullptr) {

        return nullptr;

    }

    else {

        TreeNode\* temp = createNode(root->data);

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

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

        return temp;

    }

}

void printTree(TreeNode\* root) {

    if (root == nullptr) return;

    cout << root->data << " ";

    printTree(root->left);

    printTree(root->right);

}

int main() {

    TreeNode\* root = createNode(1);

    root->left = createNode(2);

    root->right = createNode(3);

    root->left->left = createNode(4);

    root->left->right = createNode(5);

    root->right->left = createNode(6);

    root->right->right = createNode(7);

    root->left->right->left = createNode(9);

    root->right->right->left = createNode(15);

    printTree(root);

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

    cout << "Mirror tree: ";

    printTree(temp);

    return 0;

}

Покликання на пул-реквест: <https://github.com/artificial-intelligence-department/ai_programming_playground/pull/956/files#diff-79380a9a39e2c5b943be33b45ede1ba62558d7a90916892b68ea44713594a31c>

Завдання №8 Практична робота №5 “Бінарні дерева: Записати кожному батьківському вузлу суму підвузлів”

#include <iostream>

using namespace std;

struct TreeNode {

    int value;

    TreeNode\* left;

    TreeNode\* right;

};

TreeNode\* createNode(int data) {

    TreeNode\* newNode = new TreeNode();

    newNode->value = data;

    newNode->left = newNode->right = nullptr;

    return newNode;

}

void printTree(TreeNode\* root) {

    if (root == nullptr) return;

    cout << root->value << " ";

    printTree(root->left);

    printTree(root->right);

}

void tree\_sum(TreeNode \*root) {

    if (root == nullptr) {

        return;

    }

    if (root->left != nullptr) {

        tree\_sum(root->left);

        root->value += root->left->value;

    }

    if (root->right != nullptr) {

        tree\_sum(root->right);

        root->value += root->right->value;

    }

}

int main() {

    TreeNode\* root = createNode(1);

    root->left = createNode(2);

    root->right = createNode(3);

    root->left->left = createNode(4);

    root->left->right = createNode(5);

    root->right->left = createNode(6);

    root->right->right = createNode(7);

    root->left->right->left = createNode(9);

    root->right->right->left = createNode(15);

    printTree(root);

    tree\_sum(root);

    cout << "The tree with the sum of its children: ";

    printTree(root);

}

Покликання на пул-реквест: <https://github.com/artificial-intelligence-department/ai_programming_playground/pull/956/files#diff-8fc41c7e2c1b846e215a716949b7ee8db6282579a105ce50771efb30710419d3>

Завдання №9 Власна практична робота «Зуби»

#include <iostream>

using namespace std;

int sharp(int n, int k, int a[]) {

  int sharper = 0;

  for (int i = 0; i < n; i++) {

    if (a[i] >= k) {

      int length = 1;

      while (i + length < n && a[i + length] >= k) {

        length++;

      }

      sharper = max(sharper, length);

    }

  }

  return sharper;

}

int main() {

  int n, k;

  cin >> n >> k;

  int a[n];

  for (int i = 0; i < n; i++) {

    cin >> a[i];

  }

  cout << sharp(n, k, a) << endl;

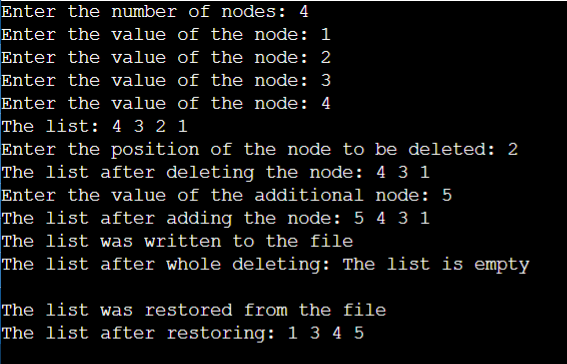
  return 0;

}

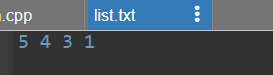
Покликання на пул-реквест: <https://github.com/artificial-intelligence-department/ai_programming_playground/pull/956/files#diff-f0093dc54c262521a88ea470bd0b9d91513adb5ec6eefa2ab2c5e12591d0b837>

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

Завдання №1 Лабораторна робота №10 "Інформаційні динамічні структури"



Малюнок : Успішна компіляція( створюється список з 4 елементів, потім видаляється елемент з індексом 2, на початок списку додається елемент зі значенням 5, список записується у файл, видаляється повністю, відновлюється через файл)



Малюнок : Вміст файлу list.txt

Затрачений час на виконання: 2.5год

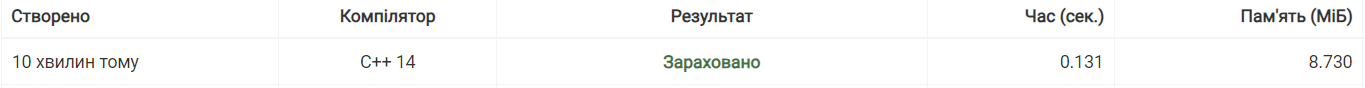
Завдання №2 Лабораторна робота №5 у Algotester



Малюнок : один із запропонованих у Algotester вводів



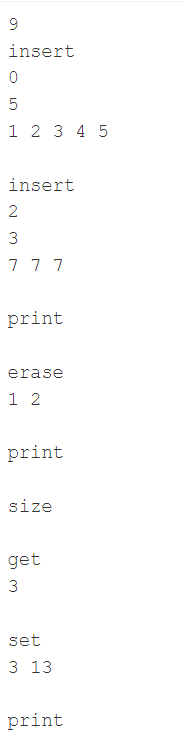
Малюнок : Відповідь співпадає із прикладом у Algotester



Малюнок : Підтвердження у Algotester

*Затрачений час на виконання: 7год*

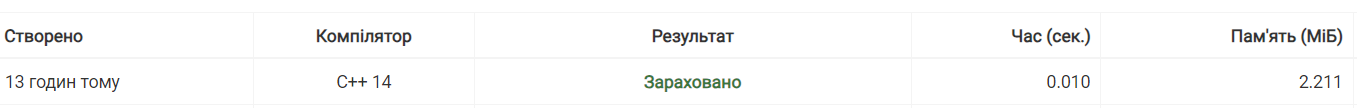
Завдання №3 Лабораторна робота №7\_8 у Algotester



Малюнок : один із запропонованих у Algotester вводів



Малюнок : Відповідь співпадає із прикладом у Algotester



Малюнок : Підтвердження у Algotester

*Затрачений час на виконання: 8год*

Завдання №4 Практична робота №1 "Зв’язаний список: Реверс списку"



Малюнок : успішна компіляція (вивело reversed list)

*Затрачений час на виконання: 1год*

Завдання №5 Практична робота №2 “Зв'язаний список: Порівняння списків”



Малюнок : успішна компіляція( списки не однакові)

*Затрачений час на виконання: 1год*

Завдання №6 Практична робота №3 “Зв'язаний список: Додавання великих чисел”



Малюнок : успішне виведення списку, що складається із суми чисел, які були створені за допомгою елементів у інших двох списках

Затрачений час на виконання: 3год

Завдання №7 Практична робота №4 “Бінарні дерева: Віддзеркалення дерева”



Малюнок : успішне виведення віддзеркаленого дерева

*Затрачений час на виконання: 2год*

Завдання №8 Практична робота №5 “Бінарні дерева: Записати кожному батьківському вузлу суму підвузлів”



Малюнок : успішно вивело дерево, батьківські вузли якого склдаються з суми підвузлів

*Затрачений час на виконання: 1.5год*

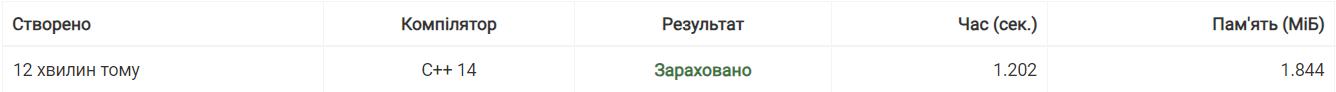
Завдання №9 Власна практична робота «Зуби»



Малюнок : один із запропонованих у Algotester вводів



Малюнок : Відповідь співпадає із прикладом у Algotester



Малюнок : Підтвердження у Algotester

*Затрачений час на виконання: 30хв*

## **6. Кооперація з командою:**

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

# 

# **Висновки:**

* Ознайомлений зі списками
* Ознайомлений з деревами
* Ознайомлений зі стеками
* Ознайомлений з чергами
* Успішно виконав лабораторні та практичні роботи