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:***

студент групи ШІ-13

Мельников Дмитро Володимирович

# **Тема роботи: *Динамічні структури (Черга, Стек, Списки, Дерево). Алгоритми обробки динамічних структур.***

# **Мета роботи: *Навчитись створювати і працювати з динамічними структурами.***

# **Теоретичні відомості:**

1. Теоретичні відомості з переліком важливих тем:

* Тема №1: Види динамічних структур
* Тема №2: Одно- та двозв’язні списки
* Тема №3: Бінарне дерево

1. Індивідуальний план опрацювання теорії:

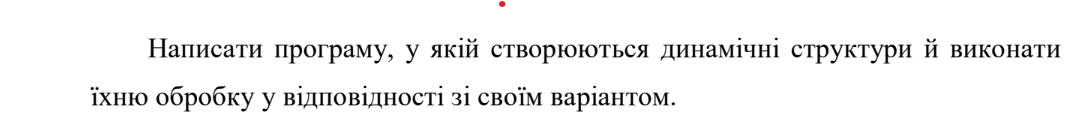
* Тема №1: Види динамічних структур
  + Джерела Інформації
    - <https://youtu.be/eSxLVD5vfqM?si=8OiPGfRKIazU-N8z>
    - <https://youtu.be/jH9ZE5uAal8?si=xdq9aK04b9Brv7Eq>
    - <https://youtu.be/5uAwaQL99cY?si=fn3dSlgevFYkswRp>
  + Що опрацьовано: Опрацьовано матеріали надані вище. Розібрано в чому різниця між різними динамічними структурами
  + Статус: Ознайомлений
  + Початок опрацювання теми: 6.12.2023
  + Звершення опрацювання теми: 14.12.2023
* Тема №2: Одно- та двозв’язні списки.
  + Джерела Інформації:
    - <https://youtu.be/N6dOwBde7-M?si=UrtPdTS4PvXcXLJc>
    - <https://youtu.be/vcQIFT79_50?si=9N_GPcCRgntvKedH>
  + Що опрацьовано: Опрацьовано матеріали надані вище. Попрактикувалась в імплементації linked list
  + Статус: Ознайомлени
  + Початок опрацювання теми: 6.12.2023
  + Звершення опрацювання теми: 14.12.2023
* Тема №3: Бінарне дерево
  + Джерела Інформації:
    - <https://youtu.be/Krr9J-_7IS0?si=A9QJWtCCxu9er9xI>
    - [https://youtu.be/qBFzNW0ALxQ?si=RgZR7\_W-PYHTQLz0](https://youtu.be/qBFzNW0ALxQ?si=RgZR7_W-PYHTQLz0%20)
    - <https://youtu.be/_IhTp8q0Mm0?si=xzi_Eb0rwYpFt3ib>
  + Що опрацьовано: Опрацьовано матеріали надані вище.
  + Статус: Ознайомлений
  + Початок опрацювання теми: 6.12.2023
  + Звершення опрацювання теми: 14.12.2023

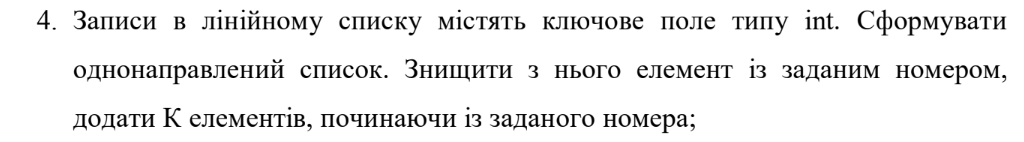
# **Виконання роботи:**

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

Завдання №1 **«Лабораторна робота №10 VNS»**

* Варіант 4





Завдання №2 **«Algotester Lab 5v3»**

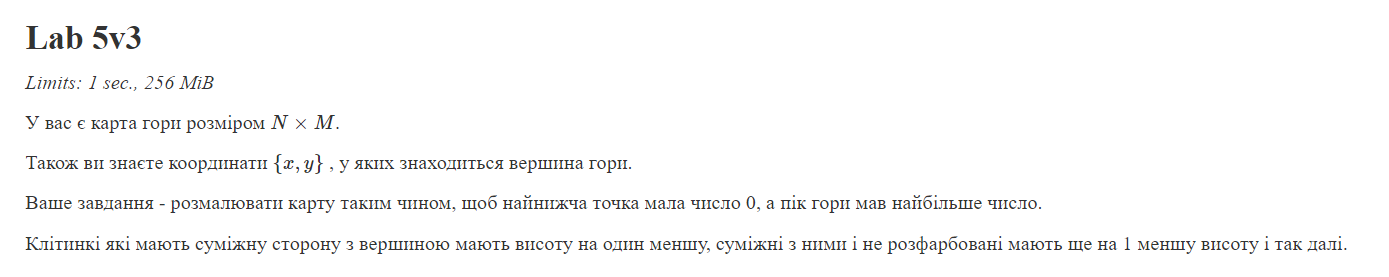
****

Рисунок algo lab 5v3

Завдання №3 **«Algotester 78v2»**

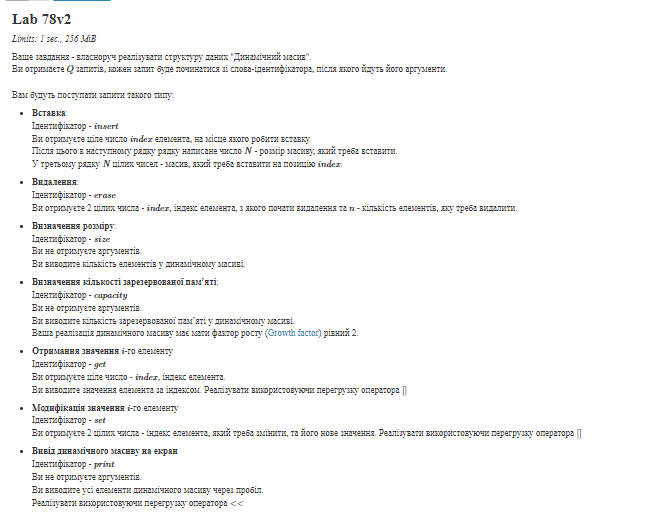
****

Рисунок algo 7v2

Завдання №4 **«Practice Task 1»**

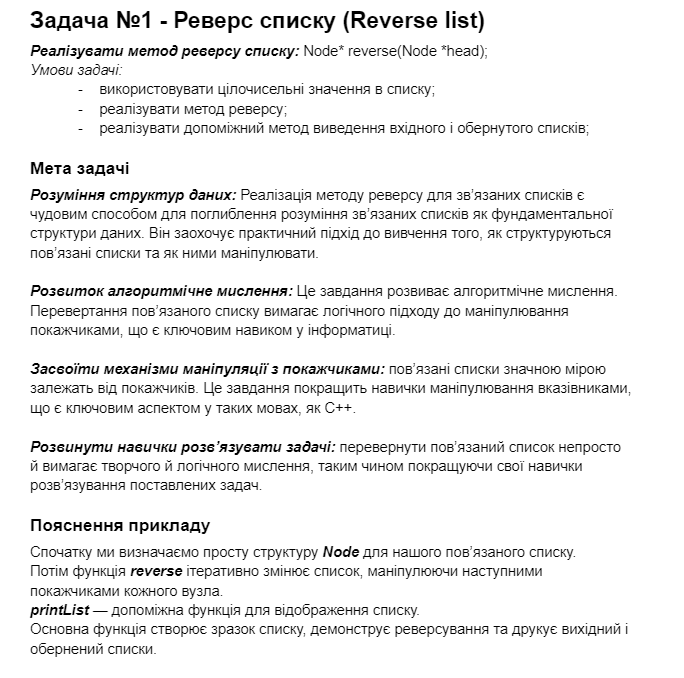
****

Рисунок practice 1

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

Програма №1 **«Лабораторна робота №10 VNS»**

* Планований час на реалізацію: до 3 годин

Програма №2 **«Algotester Lab 5v3»**

* Планований час на реалізацію : до 2 годин

Програма №3 **«Algotester 78v2»**

* Планований час на реалізацію: до 2 годин

Програма №4 **«Practice Task 1»**

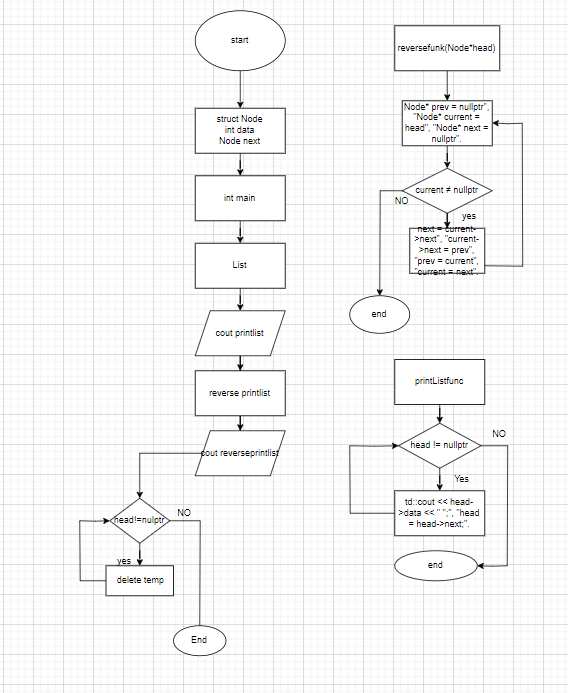
****

Рисунок diagram practice

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

Завдання №1 **«Лабораторна робота №10 VNS»**

**PR:**

#include <iostream>

#include <fstream>

struct Node {

    int key;

    Node\* next;

};

struct LinkedList {

    Node\* head;

};

// Function to create an empty linked list

void createList(LinkedList& list) {

    list.head = nullptr;

}

// Function to print the linked list

void printList(const LinkedList& list) {

    if (list.head == nullptr) {

        std::cout << "The list is empty." << std::endl;

    } else {

        Node\* current = list.head;

        std::cout << "List elements:";

        while (current != nullptr) {

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

            current = current->next;

        }

        std::cout << std::endl;

    }

}

// Function to add an item to the linked list

void addItem(LinkedList& list, int item) {

    Node\* newNode = new Node{item, nullptr};

    if (list.head == nullptr) {

        list.head = newNode;

    } else {

        Node\* current = list.head;

        while (current->next != nullptr) {

            current = current->next;

        }

        current->next = newNode;

    }

}

// Function to delete an item from the linked list

void deleteItem(LinkedList& list, int item) {

    if (list.head == nullptr) {

        std::cout << "Cannot delete from an empty list." << std::endl;

        return;

    }

    Node\* current = list.head;

    Node\* prev = nullptr;

    while (current != nullptr && current->key != item) {

        prev = current;

        current = current->next;

    }

    if (current == nullptr) {

        std::cout << "Item " << item << " not found in the list." << std::endl;

    } else {

        if (prev == nullptr) {

            list.head = current->next;

        } else {

            prev->next = current->next;

        }

        delete current;

        std::cout << "Item " << item << " deleted from the list." << std::endl;

    }

}

// Function to save the linked list to a file

void saveListToFile(const LinkedList& list, const std::string& fileName) {

    std::ofstream outFile(fileName);

    Node\* current = list.head;

    while (current != nullptr) {

        outFile << current->key << " ";

        current = current->next;

    }

    outFile.close();

    std::cout << "List saved to " << fileName << "." << std::endl;

}

// Function to delete the linked list

void deleteList(LinkedList& list) {

    Node\* current = list.head;

    while (current != nullptr) {

        Node\* next = current->next;

        delete current;

        current = next;

    }

    list.head = nullptr;

    std::cout << "List deleted." << std::endl;

}

// Function to write the linked list to a file, destroy it, and print it with a message if it's empty

void writeListToFileAndDestroy(LinkedList& list, const std::string& fileName) {

    saveListToFile(list, fileName);

    deleteList(list);

    printList(list);

}

// Function to restore the linked list from a file

void restoreListFromFile(LinkedList& list, const std::string& fileName) {

    std::ifstream inFile(fileName);

    int item;

    deleteList(list); // Clear the existing list

    while (inFile >> item) {

        addItem(list, item);

    }

    inFile.close();

    std::cout << "List restored from " << fileName << "." << std::endl;

}

int main() {

    LinkedList myList;

    createList(myList);

    printList(myList);

    addItem(myList, 42);

    addItem(myList, 10);

    addItem(myList, 8);

    printList(myList);

    deleteItem(myList, 10);

    printList(myList);

    saveListToFile(myList, "myList.txt");

    writeListToFileAndDestroy(myList, "myList2.txt");

    restoreListFromFile(myList, "myList.txt");

    printList(myList);

    deleteList(myList);

    return 0;

}

Рисунок vns 10 lab code

Завдання №2 **«Algotester Lab 5v3»**

**PR:**

#include <iostream>

#include <vector>

#include <queue>

using namespace std;

void SH(int x, int y, int N, int M, vector<vector<int>>& ar, vector<vector<bool>>& visited) { //setheight

    queue<pair<int, int>> q;

    q.push({x, y});

    visited[x][y] = true;

    ar[x][y] = 0;

    int xDir[] = {-1, 0, 1, 0};

    int yDir[] = {0, 1, 0, -1};

    while (!q.empty()) {

        pair<int, int> current = q.front();

        q.pop();

        for (int i = 0; i < 4; i++) {

            int xi = current.first + xDir[i];

            int yi = current.second + yDir[i];

            if (xi >= 0 && xi < N && yi >= 0 && yi < M && !visited[xi][yi]) {

                ar[xi][yi] = ar[current.first][current.second] + 1;

                visited[xi][yi] = true;

                q.push({xi, yi});

            }

        }

    }

}

void PG(int rws, int cls, int pkx, int pky) {//paintgrid

    vector<vector<int>> mntn(rws, vector<int>(cls, 0));

    vector<vector<bool>> visited(rws, vector<bool>(cls, false));

    SH(pkx - 1, pky - 1, rws, cls, mntn, visited);

    int maxHeight = 0;

    for (int i = 0; i < rws; i++) {

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

            maxHeight = max(maxHeight, mntn[i][j]);

        }

    }

    for (int i = 0; i < rws; i++) {

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

            cout << maxHeight - mntn[i][j] << " ";

        }

        cout << endl;

    }

}

int main() {

    int rws, cls, pkx, pky;

    cin >> rws >> cls;

    cin >> pkx >> pky;

    if (rws >= 1 && rws <= 1000 && cls >= 1 && cls <= 1000 && pkx >= 1 && pkx <= rws && pky >= 1 && pky <= cls) {

        PG(rws, cls, pkx, pky);

    } else {

        cout << "invalid input";

    }

    return 0;

}

Рисунок algo lab 5v3 code

Завдання №3 **«Algotester Lab 7v2»**

#include <iostream>

#include <string>

using namespace std;

template<typename Type>

struct MyArray {

    int size = 0;

    int capacity = 1;

    Type \*data;

    MyArray() {

        data = new Type[1];

    }

    void insert(int index, int arrSize, Type\* arr) {

        if (capacity > size + arrSize) {

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

                data[i + arrSize] = data[i];

            }

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

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

            }

            size += arrSize;

        } else {

            while (capacity <= size + arrSize) {

                capacity \*= 2;

            }

            Type \*newData = new Type[capacity];

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

                newData[i] = data[i];

            }

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

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

            }

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

                newData[i + arrSize] = data[i];

            }

            delete[] data;

            data = newData;

            size += arrSize;

        }

    }

    void erase(int index, int n) {

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

            data[i] = data[i + n];

        }

        size -= n;

    }

    int mySize() {

        return size;

    }

    int myCapacity() {

        return capacity;

    }

    Type &operator[](int index) {

        return data[index];

    }

    friend ostream &operator<<(ostream &os, const MyArray &array) {

        for (int i = 0; i < array.size; ++i) {

            os << array.data[i] << ' ';

        }

        return os;

    }

};

int main() {

    MyArray<int> myArray;

    int Q;

    cin >> Q;

    string ident;

    for (int i = 0; i < Q; ++i) {

        cin >> ident;

        if (ident == "insert") {

            int index, arrSize;

            cin >> index >> arrSize;

            int \*arr = new int[arrSize];

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

                cin >> arr[j];

            }

            myArray.insert(index, arrSize, arr);

            delete[] arr;

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

            int index, n;

            cin >> index >> n;

            myArray.erase(index, n);

        } else if (ident == "size") {

            cout << myArray.mySize() << endl;

        } else if (ident == "capacity") {

            cout << myArray.myCapacity() << endl;

        } else if (ident == "get") {

            int index;

            cin >> index;

            cout << myArray[index] << endl;

        } else if (ident == "set") {

            int index;

            cin >> index;

            cin >> myArray[index];

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

            cout << myArray << endl;

        }

    }

    return 0;

}

Рисунок algo 7v2 code

Завдання №4 **«Practice Task 1»**

**PR:**

#include <iostream>

// Структура для представлення вузла списку

struct Node {

    int data;

    Node\* next;

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

};

// Функція для реверсу списку

Node\* reverse(Node\* head) {

    Node\* prev = nullptr;

    Node\* current = head;

    Node\* next = nullptr;

    while (current != nullptr) {

        next = current->next;   // Зберігаємо наступний вузол

        current->next = prev;   // Змінюємо вказівник на наступний вузол на попередній

        prev = current;         // Переміщаємо покажчик попереднього вузла на поточний

        current = next;         // Переміщаємо покажчик поточного вузла на наступний

    }

    return prev;  // Новий початок списку - коли current стане nullptr, prev буде останнім вузлом, який тепер є початком списку

}

// Допоміжна функція для виведення списку

void printList(Node\* head) {

    while (head != nullptr) {

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

        head = head->next;

    }

    std::cout << std::endl;

}

int main() {

    // Створюємо зразок списку: 1 -> 2 -> 3 -> 4 -> 5

    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);

    std::cout << "Вихідний список: ";

    printList(head);

    // Викликаємо функцію реверсу

    head = reverse(head);

    std::cout << "Обернений список: ";

    printList(head);

    // Очищаємо пам'ять

    while (head != nullptr) {

        Node\* temp = head;

        head = head->next;

        delete temp;

    }

    return 0;

}

Рисунок practice code

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

Завдання №1 **«Лабораторна робота №10 VNS»**

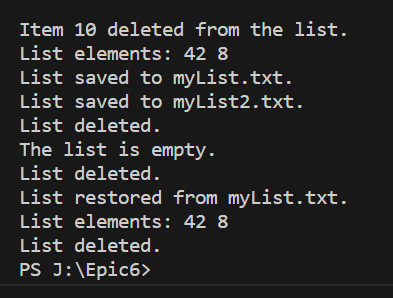


Рисунок result vns10

Час затрачений на виконання завдання: до 3 години

Завдання №2 **«Algotester Lab 5v3»**

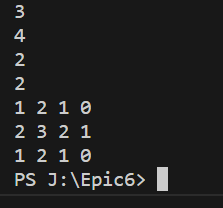


Рисунок result algo 5v3

Час затрачений на виконання завдання: до 1 години

Завдання №3 **«Algotester 78v2»**

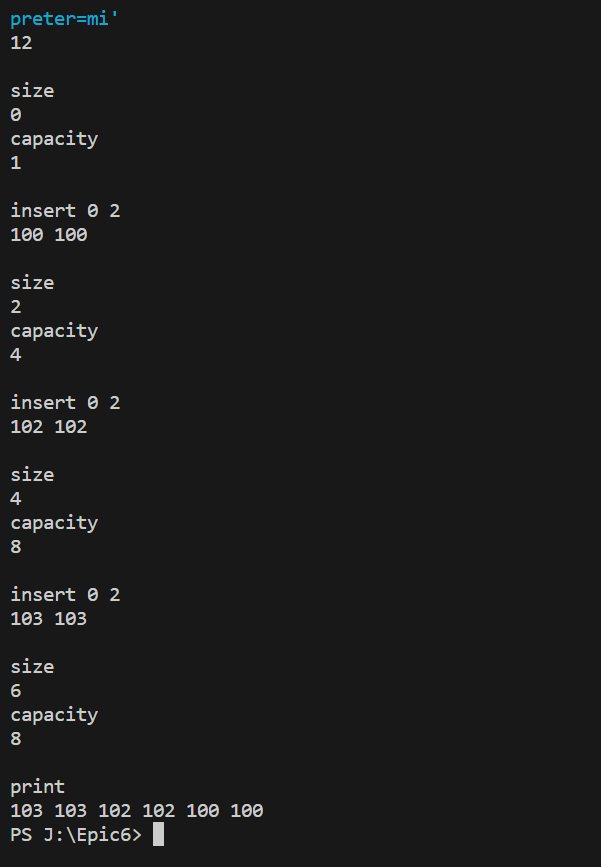


Рисунок algo result 7v2

Час затрачений на виконання завдання: близько 2 годин

Завдання №4 **«Practice Task 1»**

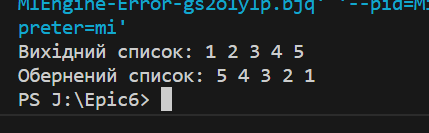
****

Рисунок practice result

Час затрачений на виконання завдання: близько години

# **Висновки:**

Під час виконання даних практичних та лабораторних робіт з епіка 6 я зміг зрозуміти основні відмінності в різних типах динамічних структур. Також вдалось попрактикуватись в реалізації зв’язних списків та бінарних дерев. Було пропрацьовано різні алгоритми обробки динамічних структур, а саме: створення списку, друк списку, вставка елемента в список, пошук по бінарному дереву, обхід дерева.