Міністерство освіти і науки України

Національний університет «Львівська політехніка»

Кафедра систем штучного інтелекту



**Звіт**

про виконання

**Лабораторних та практичних робіт № 6(Algolab 5. Algolab 78.VNS Lab 10. Class Practice Work. Self Practice Work)**

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

***з розділу***: «Епік 5: « Динамічні структури. Бінарне дерево. Динамічний масив. Лінкед ліст»

***Виконав:***

студентгрупи ШІ-13

Сенів Роман Максимович

### Тема роботи:

Динамічні структури. Бінарне дерево. Динамічний масив. Лінкед ліст

### Мета роботи:

Ознайомлення з динамічними структурами,бінарним деревом, динамічним масивов, лінкед лістом

### Теоретичні відомості:

1. Теоретичні відомості з переліком важливих тем:

* Тема №1: Динамічні структури
* Тема №2: Бінарне дерево
* Тема №3 Динамічний масив
* Тема №4 Лінкед і дабл лінкед ліст

1. Індивідуальний план опрацювання теорії:

**Тема №1 Динамічні структури**

* + Джерела Інформації

https://purecodecpp.com/uk/archives/1538

* + Що опрацьовано:

Матеріал з джерел інформації

* + Статус: Ознайомлений
  + Початок опрацювання теми: 5.12.2023
  + Звершення опрацювання теми: 5.12.2023

**Тема №2 Бінарне дерево**

* + Джерела Інформації:

https://www.bestprog.net/ru/2023/08/01/c-binary-search-tree-binarytree-class-integers-ru/#google\_vignette

* + Що опрацьовано:

Матеріал з джерел інформації

* + Статус: Ознайомлений
  + Початок опрацювання теми: 5.12.2023
  + Звершення опрацювання теми: 5.12.2023

**Тема №3 Динамічний масив**

* + Джерела Інформації

https://acode.com.ua/urok-90-dynamichni-masyvy/

Що опрацьовано:

Матеріал з джерел інформації

* + Статус: Ознайомлений
  + Початок опрацювання теми: 5.12.2023
  + Звершення опрацювання теми: 5.12.2023

**Тема №4 Лінкед і дабл лінкед ліст**

* + Джерела Інформації

https://www.bestprog.net/uk/2022/02/13/c-an-example-of-the-implementation-of-a-linear-singly-linked-list-ua/

Що опрацьовано:

Матеріал з джерел інформації

* + Статус: Ознайомлений
  + Початок опрацювання теми: 5.12.2023
  + Звершення опрацювання теми: 5.12.2023

### Виконання роботи:

**Завдання №1**

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

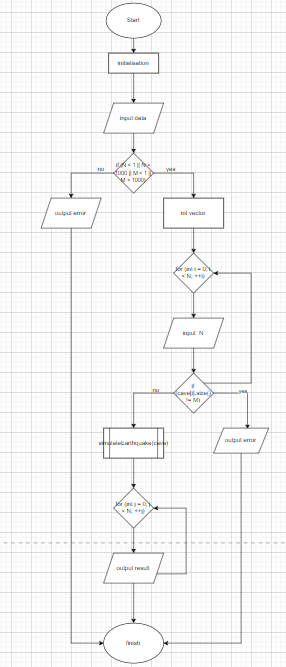
**Завдання №1**

Тут вказуєте

-Варіант 2,Algolab 5

У даній задачі, потрібно симулювати землетрус у двовимірній печері. В печері є пустота (означена символом "O"), пісок ("S") та каміння ("X"). Землетрус призводить до того, що весь пісок падає вниз до найнижчої пустої клітинки, але не може пройти через каміння. Потрібно розмістити пісок під камінням, якщо під камінням є пустота. Тобто, якщо нижня клітина каміння є пустою, пісок опиниться на цій клітинці. Процес продовжується до тих пір, поки пісок не опиниться на нижніх клітинах каміння або досягне дна печери.

1. *Дизайн та планована оцінка часу виконання завдань:*

**

1. *Код програми*

#include <iostream>

#include <vector>

using namespace std;

void simulateEarthquake(vector<string>& cave) {

    int N = cave.size();

    int M = cave[0].size();

    int col = 0;

    do {

        int empty = N - 1;

        int row = N - 1;

        do {

            if (cave[row][col] == 'X') {

                empty = row - 1;

            } else if (cave[row][col] == 'S') {

                swap(cave[row][col], cave[empty][col]);

                --empty;

            }

            --row;

        } while (row >= 0);

        ++col;

    } while (col < M);

}

int main() {

    int N, M;

    cin >> N >> M;

    if (N < 1 || N > 1000 || M < 1 || M > 1000) {

        cerr << "error" << endl;

        return 1;

    }

    vector<string> cave(N);

    int i = 0;

    do {

        cin >> cave[i];

        if (cave[i].size() != M) {

            cerr << "error" << endl;

            return 1;

        }

        ++i;

    } while (i < N);

    simulateEarthquake(cave);

    int j = 0;

    do {

        cout << cave[j] << endl;

        ++j;

    } while (j < N);

    return 0;

}

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

****

Потрачено часу: 1 день

**Завдання №2**

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

-Варіант 1,Algolab 78

1. *Код програми*

#include <iostream>

template <typename T>

class Node {

public:

    T data;

    Node\* prev;

    Node\* next;

    Node(const T& value) : data(value), prev(nullptr), next(nullptr) {}

};

template <typename T>

class DoublyLinkedList {

private:

    Node<T>\* head;

    Node<T>\* tail;

    int size;

public:

    DoublyLinkedList() : head(nullptr), tail(nullptr), size(0) {}

    void insert(int index, int N, const T\* values) {

        if (index < 0 || index > size) {

            return;

        }

        Node<T>\* newNode = nullptr;

        for (int i = 0; i < N; ++i) {

            newNode = new Node<T>(values[i]);

            if (head == nullptr) {

                head = tail = newNode;

            } else if (index == 0) {

                newNode->next = head;

                head->prev = newNode;

                head = newNode;

            } else if (index == size) {

                tail->next = newNode;

                newNode->prev = tail;

                tail = newNode;

            } else {

                Node<T>\* current = head;

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

                    current = current->next;

                }

                newNode->next = current->next;

                newNode->prev = current;

                current->next->prev = newNode;

                current->next = newNode;

            }

            ++size;

            ++index;

        }

    }

    void erase(int index, int n) {

        if (index < 0 || index >= size || n <= 0) {

            return;

        }

        Node<T>\* current = head;

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

            current = current->next;

        }

        while (n > 0 && current != nullptr) {

            Node<T>\* nextNode = current->next;

            if (current->prev != nullptr) {

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

            } else {

                head = current->next;

            }

            if (current->next != nullptr) {

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

            } else {

                tail = current->prev;

            }

            delete current;

            current = nextNode;

            --size;

            --n;

        }

    }

    int getSize() const {

        return size;

    }

    T get(int index) const {

        if (index < 0 || index >= size) {

            return T();

        }

        Node<T>\* current = head;

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

            current = current->next;

        }

        return current->data;

    }

    void set(int index, const T& value) {

        if (index < 0 || index >= size) {

            return;

        }

        Node<T>\* current = head;

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

            current = current->next;

        }

        current->data = value;

    }

    friend std::ostream& operator<<(std::ostream& os, const DoublyLinkedList<T>& list) {

        Node<T>\* current = list.head;

        while (current != nullptr) {

            os << current->data << " ";

            current = current->next;

        }

        return os;

    }

};

int main() {

    int Q;

    std::cin >> Q;

    DoublyLinkedList<int> list;

    for (int i = 0; i < Q; ++i) {

        std::string operation;

        std::cin >> operation;

        if (operation == "insert") {

            int index, N;

            std::cin >> index >> N;

            int\* values = new int[N];

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

                std::cin >> values[j];

            }

            list.insert(index, N, values);

            delete[] values;

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

            int index, n;

            std::cin >> index >> n;

            list.erase(index, n);

        } else if (operation == "size") {

            std::cout << list.getSize() << std::endl;

        } else if (operation == "get") {

            int index;

            std::cin >> index;

            std::cout << list.get(index) << std::endl;

        } else if (operation == "set") {

            int index, value;

            std::cin >> index >> value;

            list.set(index, value);

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

            std::cout << list << std::endl;

        }

    }

    return 0;

}

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



Потрачено часу: 1 день

**Завдання №3**

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

-Варіант 5,VNS Lab 10

Записи в лінійному списку містять ключове поле типу int. Сформувати

однонаправлений список. Знищити з нього К елементів, починаючи із

заданого номера, додати К елементів, починаючи із заданого номера;

1. *Код програми*

#include <iostream>

#include <fstream>

#include <vector>

class MyListHandler {

private:

    std::vector<int> myVector;

public:

    void createList() {

        myVector.clear();

    }

    void addElement(int element) {

        myVector.push\_back(element);

    }

    void removeElement(int element) {

        auto it = std::find(myVector.begin(), myVector.end(), element);

        if (it != myVector.end()) {

            myVector.erase(it);

        } else {

            std::cout << "Element " << element << " not found in the list." << std::endl;

        }

    }

    void printList() {

        std::cout << "Current list: ";

        for (int element : myVector) {

            std::cout << element << " ";

        }

        std::cout << std::endl;

    }

    void writeToFile(const std::string& filename) {

        std::ofstream outFile(filename, std::ios::binary);

        if (outFile.is\_open()) {

            for (int element : myVector) {

                outFile.write(reinterpret\_cast<const char\*>(&element), sizeof(element));

            }

            std::cout << "List has been written to " << filename << "." << std::endl;

        } else {

            std::cout << "Unable to open file: " << filename << std::endl;

        }

    }

    void destroyList() {

        myVector.clear();

        std::cout << "List has been destroyed." << std::endl;

    }

    void restoreFromFile(const std::string& filename) {

        std::ifstream inFile(filename, std::ios::binary);

        if (inFile.is\_open()) {

            myVector.clear();

            int element;

            while (inFile.read(reinterpret\_cast<char\*>(&element), sizeof(element))) {

                myVector.push\_back(element);

            }

            std::cout << "List has been restored from " << filename << "." << std::endl;

        } else {

            std::cout << "Unable to open file: " << filename << std::endl;

        }

    }

};

int main() {

    MyListHandler myListHandler;

    myListHandler.createList();

    myListHandler.addElement(1);

    myListHandler.addElement(2);

    myListHandler.addElement(3);

    myListHandler.printList();

    myListHandler.writeToFile("my\_list.bin");

    myListHandler.destroyList();

    myListHandler.printList();

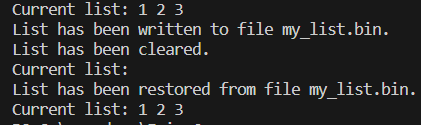
    myListHandler.restoreFromFile("my\_list.bin");

    myListHandler.printList();

    return 0;

}

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

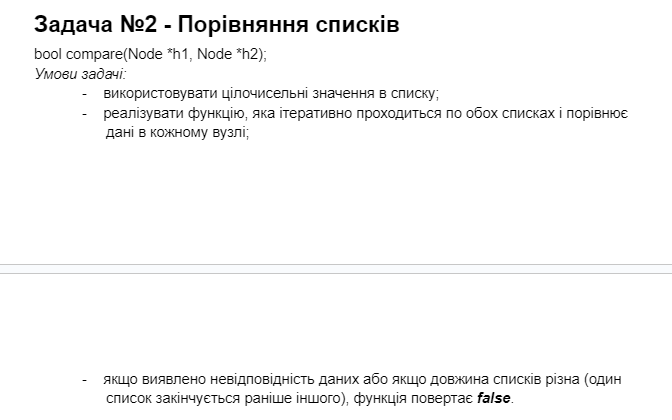


Потрачено часу: 1 день

**Завдання №4**

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

- Class Practice Work



1. *Код програми*

#include <iostream>

struct Node {

    int data;

    Node\* next;

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

};

Node\* reverse(Node\* head) {

    Node\* prev = nullptr;

    while (head) {

        Node\* next = head->next;

        head->next = prev;

        prev = head;

        head = next;

    }

    return prev;

}

void printList(Node\* head) {

    while (head) {

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

        head = head->next;

    }

    std::cout << std::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);

    std::cout << "Input list: ";

    printList(head);

    head = reverse(head);

    std::cout << "Reversed list: ";

    printList(head);

    while (head) {

        Node\* temp = head;

        head = head->next;

        delete temp;

    }

    return 0;

}

**

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

### Висновки:

Ознайомлений з динамічними структурами,бінарним деревом, динамічним масивов, лінкед лістом.