**Ministerul Educaţiei și Cercetării**

**Universitatea Tehnică a Moldovei**

**Faculatea Calculatoare, Informatică și Microelectronică**

**Departamentul Ingineria Software și Automatică**



**RAPORT**

**Lucrare de Laborator nr.1**

**SDA**

A efectuat:

st. gr. CR-231 Maletchi Mirela

A verificat: Vladina Tutunaru

asist. univ

**Chişinău 2024**

**main.cpp**

#include "Header.h"

int main() {

    int optiune;

    do {

        cout << "Meniu:" << endl;

        cout << "1. Alocare dinamica a memoriei pentru tabloul de structuri." << endl;

        cout << "2. Introducerea elementelor tabloului de la tastatura." << endl;

        cout << "3. Cautarea elementului tabloului." << endl;

        cout << "4. Sortarea tabloului." << endl;

        cout << "5. Redactarea elementului tabloului." << endl;

        cout << "6. Adaugarea unui element nou la sfarsit." << endl;

        cout << "7. Eliminarea elementului indicat din tablou." << endl;

        cout << "8. Inserarea unui element nou." << endl;

        cout << "9. Salvarea elementelor tabloului in fisier." << endl;

        cout << "10. Citirea elementelor tabloului din fisier." << endl;

        cout << "11. Extragerea elementelor tabloului la ecran." << endl;

        cout << "12. Eliberarea memoriei alocate pentru tablou." << endl;

        cout << "0. Iesire din program." << endl;

        cout << "Alegeti o optiune: ";

        cin >> optiune;

        switch (optiune) {

            case 1:

                alocare\_dinamica();

                break;

            case 2:

                introducere\_elemente();

                break;

            case 3:

                cautare\_element();

                break;

            case 4:

                sortare\_tablou();

                break;

            case 5:

                redactare\_element();

                break;

            case 6:

                adaugare\_element();

                break;

            case 7:

                eliminare\_element();

                break;

            case 8:

                inserare\_element();

                break;

            case 9:

                salvare\_in\_fisier();

                break;

            case 10:

                citire\_din\_fisier();

                break;

            case 11:

                afisare\_elemente();

                break;

            case 12:

                eliberare\_memorie();

                break;

            case 0:

                break;

            default:

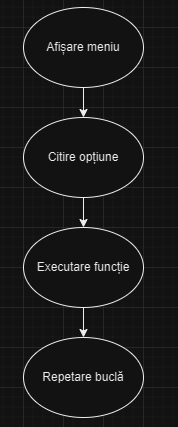
                cout << "Optiune invalida. Alegeti din nou." << endl;

        }

    } while (optiune != 0);

    return 0;

}

****

**Fig.1 schema logica pentru main.cpp**

**Header.h**

#ifndef HEADER\_H

#define HEADER\_H

#include <iostream>

#include <fstream>

#include <cstring>

#include <algorithm>

using namespace std;

struct Stat {

    char denumire[100];

    char capitala[50];

    float suprafata;

    int populatie;

    float pib;

};

extern Stat \*tablou;

extern int dimensiune\_tablou;

void alocare\_dinamica();

void introducere\_elemente();

void cautare\_element();

void sortare\_tablou();

void redactare\_element();

void adaugare\_element();

void eliminare\_element();

void inserare\_element();

void salvare\_in\_fisier();

void citire\_din\_fisier();

void afisare\_elemente();

void eliberare\_memorie();

int comparare\_stat(const void \*a, const void \*b);

#endif /\* HEADER\_H \*/

****

**Fig.2 schema logica pentru Header.h**

**Functii.cpp**

#include "Header.h"

Stat \*tablou = nullptr;

int dimensiune\_tablou = 0;

void alocare\_dinamica() {

    if (tablou != nullptr) {

        cout << "Memoria a fost deja alocata." << endl;

    }

    cout << "Introduceti dimensiunea tabloului: ";

    cin >> dimensiune\_tablou;

    tablou = new Stat[dimensiune\_tablou];

    if (tablou == nullptr) {

        cout << "Eroare la alocarea dinamica a memoriei." << endl;

    }

    cout << "Alocare reusita." << endl;

}

void introducere\_elemente() {

    if (tablou == nullptr) {

        cout << "Alocati memorie pentru tablou inainte de introducere." << endl;

    }

    for (int i = 0; i < dimensiune\_tablou; ++i) {

        cout << "Introduceti detalii pentru Statul " << i + 1 << ":" << endl;

        cout << "Denumire: ";

        cin >> tablou[i].denumire;

        cout << "Capitala: ";

        cin >> tablou[i].capitala;

        cout << "Suprafata: ";

        cin >> tablou[i].suprafata;

        cout << "Populatie: ";

        cin >> tablou[i].populatie;

        cout << "PIB: ";

        cin >> tablou[i].pib;

    }

}

void cautare\_element() {

    if (tablou == nullptr) {

        cout << "Alocati memorie pentru tablou inainte de cautare." << endl;

    }

    char denumire\_cautata[100];

    cout << "Introduceti denumirea statului cautat: ";

    cin >> denumire\_cautata;

    bool gasit = false;

    for (int i = 0; i < dimensiune\_tablou; ++i) {

        if (strcmp(tablou[i].denumire, denumire\_cautata) == 0) {

            cout << "Statul a fost gasit:" << endl;

            cout << "Denumire: " << tablou[i].denumire << endl;

            cout << "Capitala: " << tablou[i].capitala << endl;

            cout << "Suprafata: " << tablou[i].suprafata << endl;

            cout << "Populatie: " << tablou[i].populatie << endl;

            cout << "PIB: " << tablou[i].pib << endl;

            gasit = true;

            break;

        }

    }

    if (!gasit) {

        cout << "Statul cu denumirea '" << denumire\_cautata << "' nu a fost gasit." << endl;

    }

}

void sortare\_tablou() {

    if (tablou == nullptr) {

        cout << "Alocati memorie pentru tablou inainte de sortare." << endl;

        return;

    }

    qsort(tablou, dimensiune\_tablou, sizeof(Stat), comparare\_stat);

    cout << "Tabloul a fost sortat." << endl;

}

void redactare\_element() {

    if (tablou == nullptr) {

        cout << "Alocati memorie pentru tablou inainte de redactare." << endl;

        return;

    }

    int index;

    cout << "Introduceti indexul statului de redactat: ";

    cin >> index;

    if (index < 1 || index > dimensiune\_tablou) {

        cout << "Index invalid." << endl;

        return;

    }

    cout << "Introduceti noile detalii pentru Statul " << index << ":" << endl;

    cout << "Denumire: ";

    cin >> tablou[index - 1].denumire;

    cout << "Capitala: ";

    cin >> tablou[index - 1].capitala;

    cout << "Suprafata: ";

    cin >> tablou[index - 1].suprafata;

    cout << "Populatie: ";

    cin >> tablou[index - 1].populatie;

    cout << "PIB: ";

    cin >> tablou[index - 1].pib;

}

void adaugare\_element() {

    if (tablou == nullptr) {

        cout << "Alocati memorie pentru tablou inainte de adaugare." << endl;

        return;

    }

    dimensiune\_tablou++;

    tablou = new Stat[dimensiune\_tablou];

    if (tablou == nullptr) {

        cout << "Eroare la realocarea memoriei." << endl;

        return;

    }

    cout << "Introduceti detalii pentru Statul " << dimensiune\_tablou << ":" << endl;

    cout << "Denumire: ";

    cin >> tablou[dimensiune\_tablou - 1].denumire;

    cout << "Capitala: ";

    cin >> tablou[dimensiune\_tablou - 1].capitala;

    cout << "Suprafata: ";

    cin >> tablou[dimensiune\_tablou - 1].suprafata;

    cout << "Populatie: ";

    cin >> tablou[dimensiune\_tablou - 1].populatie;

    cout << "PIB: ";

    cin >> tablou[dimensiune\_tablou - 1].pib;

}

void eliminare\_element() {

    if (tablou == nullptr) {

        cout << "Alocati memorie pentru tablou inainte de eliminare." << endl;

        return;

    }

    int index;

    cout << "Introduceti indexul statului de eliminat: ";

    cin >> index;

    if (index < 1 || index > dimensiune\_tablou) {

        cout << "Index invalid." << endl;

        return;

    }

    for (int i = index - 1; i < dimensiune\_tablou - 1; ++i) {

        tablou[i] = tablou[i + 1];

    }

    dimensiune\_tablou--;

    Stat \*nou\_tablou = new Stat[dimensiune\_tablou];

    for (int i = 0; i < dimensiune\_tablou; ++i) {

        nou\_tablou[i] = tablou[i];

    }

    delete[] tablou;

    tablou = nou\_tablou;

    cout << "Statul de la indexul " << index << " a fost eliminat." << endl;

}

void inserare\_element() {

    if (tablou == nullptr) {

        cout << "Alocati memorie pentru tablou inainte de inserare." << endl;

        return;

    }

    int index;

    cout << "Introduceti indexul la care sa fie inserat noul stat: ";

    cin >> index;

    if (index < 1 || index > dimensiune\_tablou + 1) {

        cout << "Index invalid." << endl;

        return;

    }

    dimensiune\_tablou++;

    Stat \*nou\_tablou = new Stat[dimensiune\_tablou];

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

        nou\_tablou[i] = tablou[i];

    }

    cout << "Introduceti detalii pentru noul stat:" << endl;

    cout << "Denumire: ";

    cin >> nou\_tablou[index - 1].denumire;

    cout << "Capitala: ";

    cin >> nou\_tablou[index - 1].capitala;

    cout << "Suprafata: ";

    cin >> nou\_tablou[index - 1].suprafata;

    cout << "Populatie: ";

    cin >> nou\_tablou[index - 1].populatie;

    cout << "PIB: ";

    cin >> nou\_tablou[index - 1].pib;

    for (int i = index - 1; i < dimensiune\_tablou; ++i) {

        nou\_tablou[i + 1] = tablou[i];

    }

    delete[] tablou;

    tablou = nou\_tablou;

    cout << "Statul a fost inserat la indexul " << index << "." << endl;

}

void salvare\_in\_fisier() {

    if (tablou == nullptr) {

        cout << "Alocati memorie pentru tablou inainte de salvare." << endl;

        return;

    }

    ofstream fisier("stat.txt");

    if (!fisier) {

        cout << "Eroare la deschiderea fisierului pentru scriere." << endl;

        return;

    }

    for (int i = 0; i < dimensiune\_tablou; ++i) {

        fisier << tablou[i].denumire << " " << tablou[i].capitala << " " << tablou[i].suprafata << " "

               << tablou[i].populatie << " " << tablou[i].pib << endl;

    }

    fisier.close();

    cout << "Datele au fost salvate in fisierul stat.txt." << endl;

}

void citire\_din\_fisier() {

    ifstream fisier("stat.txt");

    if (!fisier) {

        cout << "Eroare la deschiderea fisierului pentru citire." << endl;

        return;

    }

    while (fisier >> tablou[dimensiune\_tablou].denumire >> tablou[dimensiune\_tablou].capitala

           >> tablou[dimensiune\_tablou].suprafata >> tablou[dimensiune\_tablou].populatie

           >> tablou[dimensiune\_tablou].pib) {

        dimensiune\_tablou++;

        Stat \*nou\_tablou = new Stat[dimensiune\_tablou];

        for (int i = 0; i < dimensiune\_tablou - 1; ++i) {

            nou\_tablou[i] = tablou[i];

        }

        delete[] tablou;

        tablou = nou\_tablou;

    }

    fisier.close();

    cout << "Datele au fost citite din fisierul stat.txt." << endl;

}

void afisare\_elemente() {

    if (tablou == nullptr) {

        cout << "Alocati memorie pentru tablou inainte de afisare." << endl;

        return;

    }

    for (int i = 0; i < dimensiune\_tablou; ++i) {

        cout << "Statul " << i + 1 << ":" << endl;

        cout << "Denumire: " << tablou[i].denumire << endl;

        cout << "Capitala: " << tablou[i].capitala << endl;

        cout << "Suprafata: " << tablou[i].suprafata << endl;

        cout << "Populatie: " << tablou[i].populatie << endl;

        cout << "PIB: " << tablou[i].pib << endl;

    }

}

void eliberare\_memorie() {

    delete[] tablou;

    tablou = nullptr;

    dimensiune\_tablou = 0;

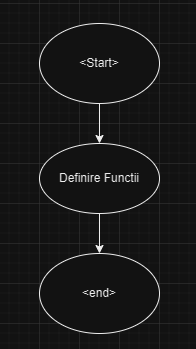
    cout << "Memoria a fost eliberata." << endl;

}

int comparare\_stat(const void \*a, const void \*b) {

    return strcmp(((Stat \*)a)->denumire, ((Stat \*)b)->denumire);

}

****

**Fig.3 schema logica pentru Functii.cpp**

**Concluzie:**

În cadrul acestui laborator, am dezvoltat o aplicație în limbajul de programare C++ pentru gestionarea informațiilor despre diferite state. Acest proces a implicat crearea a trei componente principale: fișierul Header.h, fișierul Functii.cpp și fișierul main.cpp. prin această abordare structurată și modulară, am reușit să dezvolt o aplicație eficientă pentru gestionarea informațiilor despre state, utilizând limbajul de programare C++. Această aplicație oferă o modalitate convenabilă pentru manipularea datelor despre state și poate fi extinsă și adaptată în funcție de nevoile specifice ale utilizatorului sau ale proiectului.