**Ministerul Educaţiei și Cercetării**

**Universitatea Tehnică a Moldovei**

**Faculatea Calculatoare, Informatică și Microelectronică**

**Departamentul Ingineria Software și Automatică**



**RAPORT**

**Lucrare de Laborator nr.2**

**SDA**

**Tema: Implementarea tipului abstract de date „Listă**

**unidirecțională” („Listă simplu înlănțuită”) în limbajul C. Partea I.**

A efectuat:

st. gr. CR-231 Maletchi Mirela

A verificat: Vladina Tutunaru

asist. univ

**Chişinău 2024**

**Scopul lucrării:** Obţinerea deprinderilor practice de implementare și de utilizare a tipului abstract de date (TAD) “Listă simplu înlănţuită” în limbajul C cu asigurarea operațiilor de prelucrare de bază ale listei.

**Sarcină:** Să se scrie trei fişiere-text în limbajul C pentru implementarea și utilizarea TAD “Listă simplu înlănţuită” cu asigurarea operațiilor de prelucrare de bază ale listei:

1. Fişier antet cu extensia .h, care conține specificarea structurii de date a elementului listei simplu înlănțuite (conform variantelor) şi prototipurile funcţiilor de prelucrare de bază ale listei.

2. Fişier cu extensia .cpp sau .c, care conține implementările (codurile) funcţiilor declarate în fişierul antet.

3. Fişier al utilizatorului, funcţia mаin() pentru prelucrarea listei cu afişarea la ecran a următorului meniu de opţiuni de bază:

1. Crearea listei în memoria dinamică.

2. Introducerea informației despre elementele listei de la tastatură.

3. Afişarea informației despre elementele listei la ecran.

4. Căutarea elementului în listă.

5. Modificarea câmpurilor unui element din listă.

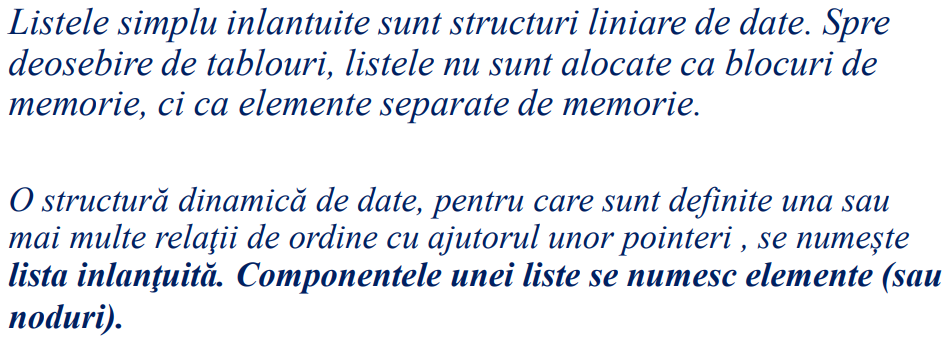
6. Determinarea adresei ultimului element din listă.

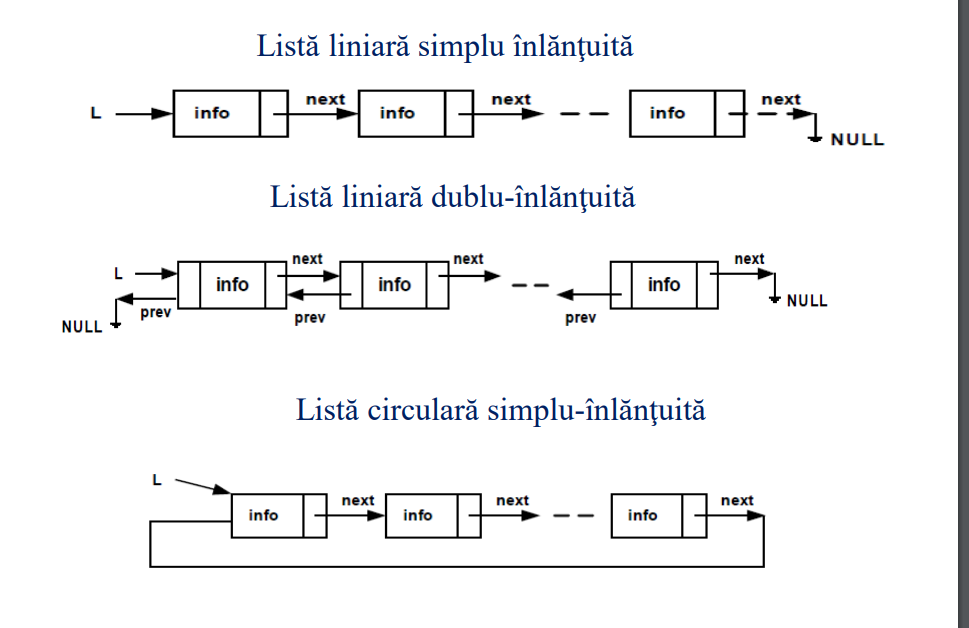
7. Determinarea lungimii listei (numărul de elemente).

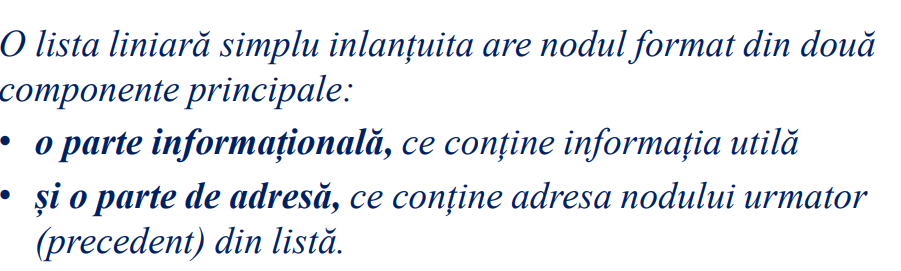
8. Înterschimbarea a două elemente indicate în listă.

9. Sortarea listei. 10. Eliberarea memoriei alocate pentru listă. 0. Ieşire din program

**Mersul lucrarii:**

****

****

****

**main.cpp**

#include <iostream>

#include <cstring>

#include "lista.h"

using namespace std;

int main() {

    Nod\* lista = nullptr;

    int optiune;

    do {

        cout << endl << "Meniu:" << endl;

        cout << "1. Crearea listei în memoria dinamică." << endl;

        cout << "2. Introducerea informației despre țările listei de la tastatură." << endl;

        cout << "3. Afișarea informației despre țările listei la ecran." << endl;

        cout << "4. Căutarea unei țări în listă." << endl;

        cout << "5. Modificarea informațiilor despre o țară din listă." << endl;

        cout << "6. Afișarea adresei primei țări din listă." << endl; // Am schimbat textul

        cout << "7. Determinarea lungimii listei (numărul de țări)." << endl;

        cout << "8. Înterschimbarea a două țări indicate în listă." << endl;

        cout << "9. Sortarea listei." << endl;

        cout << "10. Eliberarea memoriei alocate pentru listă." << endl;

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

        cout << "Introduceți opțiunea: ";

        cin >> optiune;

        switch (optiune) {

            case 1: {

                eliberareMemorie(lista);

                lista = creareLista();

                break;

            }

            case 2: {

                Tara tara;

                cout << "Introduceți detaliile pentru noua țară:" << endl;

                cout << "Statul: ";

                cin >> tara.statul;

                cout << "Capitala: ";

                cin >> tara.capitala;

                cout << "Numărul de locuitori: ";

                cin >> tara.numarLocuitori;

                cout << "Suprafața: ";

                cin >> tara.suprafata;

                cout << "PIB: ";

                cin >> tara.PIB;

                adaugaElement(lista, tara);

                break;

            }

            case 3:

                afiseazaLista(lista);

                break;

            case 4: {

                char statul[50];

                cout << "Introduceți statul pentru căutare: ";

                cin >> statul;

                if (cautaElement(lista, statul)) {

                    cout << "Țara a fost găsită." << endl;

                } else {

                    cout << "Țara nu a fost găsită." << endl;

                }

                break;

            }

            case 5: {

                char statul[50];

                cout << "Introduceți statul pentru modificare: ";

                cin >> statul;

                modificaElement(lista, statul);

                break;

            }

            case 6: {

                if (lista != nullptr) {

                    cout << "Adresa primei țări din listă: " << static\_cast<void\*>(lista) << endl;

                } else {

                    cout << "Lista este goală." << endl;

                }

                break;

            }

            case 7:

                cout << "Lungimea listei este: " << lungimeLista(lista) << endl;

                break;

            case 8: {

                char statul1[50], statul2[50];

                cout << "Introduceți statul primei țări: ";

                cin >> statul1;

                cout << "Introduceți statul celei de-a doua țări: ";

                cin >> statul2;

                interschimbaElemente(lista, statul1, statul2);

                break;

            }

            case 9:

                sortareLista(lista);

                cout << "Lista a fost sortată." << endl;

                break;

            case 10:

                eliberareMemorie(lista);

                cout << "Memoria alocată listei a fost eliberată." << endl;

                break;

            case 0:

                eliberareMemorie(lista);

                cout << "Programul s-a încheiat." << endl;

                break;

            default:

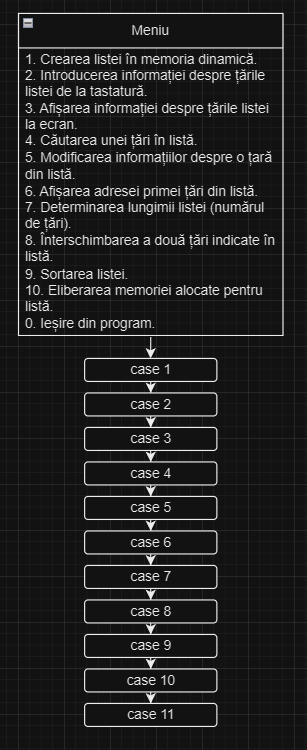
                cout << "Opțiune invalidă. Încercați din nou." << endl;

        }

    } while (optiune != 0);

    return 0;

}

****

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

**lista.h**

#ifndef lista\_h

#define lista\_h

struct Tara {

    char statul[50];

    char capitala[50];

    int numarLocuitori;

    float suprafata;

    float PIB;

};

struct Nod {

    Tara info;

    Nod\* urmator;

};

Nod\* creareLista();

void adaugaElement(Nod\*& primul, Tara elem);

void afiseazaLista(Nod\* primul);

bool cautaElement(Nod\* primul, char\* statul);

void modificaElement(Nod\* primul, char\* statul);

Nod\* adresaUltimuluiElement(Nod\* primul);

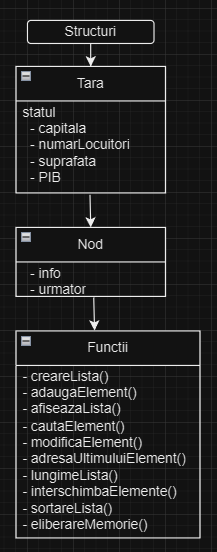
int lungimeLista(Nod\* primul);

void interschimbaElemente(Nod\* primul, char\* statul1, char\* statul2);

void sortareLista(Nod\*& primul);

void eliberareMemorie(Nod\*& primul);

#endif

****

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

**Lista.cpp**

#include "lista.h"

#include <iostream>

#include <cstring>

using namespace std;

Nod\* creareLista() {

    return nullptr;

}

void adaugaElement(Nod\*& primul, Tara elem) {

    Nod\* nou = new Nod;

    nou->info = elem;

    nou->urmator = nullptr;

    if (primul == nullptr) {

        primul = nou;

    } else {

        Nod\* temp = primul;

        while (temp->urmator != nullptr) {

            temp = temp->urmator;

        }

        temp->urmator = nou;

    }

}

void afiseazaLista(Nod\* primul) {

    if (primul == nullptr) {

        cout << "Lista este goală." << endl;

        return;

    }

    Nod\* temp = primul;

    while (temp != nullptr) {

        cout << "Statul: " << temp->info.statul << endl;

        cout << "Capitala: " << temp->info.capitala << endl;

        cout << "Numărul de locuitori: " << temp->info.numarLocuitori << endl;

        cout << "Suprafața: " << temp->info.suprafata << endl;

        cout << "PIB: " << temp->info.PIB << endl << endl;

        temp = temp->urmator;

    }

}

bool cautaElement(Nod\* primul, char\* statul) {

    Nod\* temp = primul;

    while (temp != nullptr) {

        if (strcmp(temp->info.statul, statul) == 0) {

            return true;

        }

        temp = temp->urmator;

    }

    return false;

}

void modificaElement(Nod\* primul, char\* statul) {

    Nod\* temp = primul;

    while (temp != nullptr) {

        if (strcmp(temp->info.statul, statul) == 0) {

            cout << "Introduceți noile detalii pentru statul " << statul << ":" << endl;

            cout << "Capitala: ";

            cin >> temp->info.capitala;

            cout << "Numărul de locuitori: ";

            cin >> temp->info.numarLocuitori;

            cout << "Suprafața: ";

            cin >> temp->info.suprafata;

            cout << "PIB: ";

            cin >> temp->info.PIB;

            return;

        }

        temp = temp->urmator;

    }

    cout << "Statul " << statul << " nu a fost găsit." << endl;

}

Nod\* adresaUltimuluiElement(Nod\* primul) {

    if (primul == nullptr)

        return nullptr;

    Nod\* temp = primul;

    while (temp->urmator != nullptr) {

        temp = temp->urmator;

    }

    return temp;

}

int lungimeLista(Nod\* primul) {

    int lungime = 0;

    Nod\* temp = primul;

    while (temp != nullptr) {

        lungime++;

        temp = temp->urmator;

    }

    return lungime;

}

void interschimbaElemente(Nod\* primul, char\* statul1, char\* statul2) {

    Nod\* nod1 = nullptr;

    Nod\* nod2 = nullptr;

    Nod\* temp = primul;

    while (temp != nullptr) {

        if (strcmp(temp->info.statul, statul1) == 0) {

            nod1 = temp;

        }

        if (strcmp(temp->info.statul, statul2) == 0) {

            nod2 = temp;

        }

        temp = temp->urmator;

    }

    if (nod1 != nullptr && nod2 != nullptr) {

        Tara tempInfo = nod1->info;

        nod1->info = nod2->info;

        nod2->info = tempInfo;

        cout << "Țările au fost interschimbate." << endl;

    } else {

        cout << "Una sau ambele țări nu au fost găsite." << endl;

    }

}

void sortareLista(Nod\*& primul) {

    if (primul == nullptr || primul->urmator == nullptr) {

        return;

    }

    Nod\* current = primul;

    while (current != nullptr) {

        Nod\* temp = current->urmator;

        while (temp != nullptr) {

            if (strcmp(current->info.statul, temp->info.statul) > 0) {

                Tara tempInfo = current->info;

                current->info = temp->info;

                temp->info = tempInfo;

            }

            temp = temp->urmator;

        }

        current = current->urmator;

    }

}

void eliberareMemorie(Nod\*& primul) {

    while (primul != nullptr) {

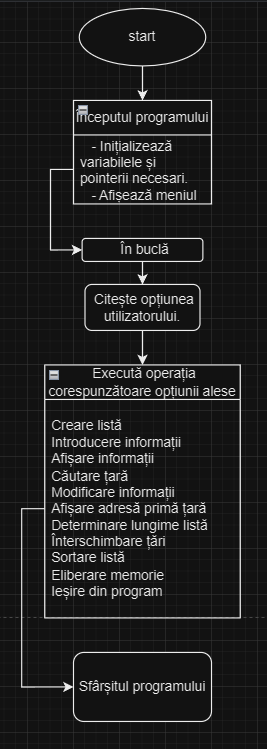
        Nod\* temp = primul;

        primul = primul->urmator;

        delete temp;

    }

}

****

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

**Concluzie:**

În urma acestui laborator, s-a constatat că organizarea codului în clase și metode facilitează dezvoltarea și menținerea proiectelor individuale. Implementarea corectă a funcționalităților și gestionarea memoriei sunt aspecte esențiale pentru prevenirea erorilor și asigurarea performanței optime a aplicațiilor. Este important să se acorde atenție comunicării cu alți dezvoltatori și să se utilizeze resursele disponibile pentru a depăși obstacolele întâlnite în timpul dezvoltării individuale.

**Sursele:**

*https://else.fcim.utm.md/pluginfile.php/131974/mod\_resource/content/0/Liste\_.pdf*