

**MINISTERUL EDUCAȚIEI, CULTURII ȘI CERCETĂRII**

**AL REPUBLICII MOLDOVA**

**Universitatea Tehnică a Moldovei**

**Facultatea Calculatoare, Informatică şi Microelectronică**

**Departamentul Informatică şi Ingineria Sistemelor**

**gr. IA-233, Chistol Maxim**

**Raport**

**pentru lucrarea de laborator Nr.7**

***la cursul de “Structuri de Date si Algoritmi”***

Verificat:

**Maria Guțu,** *Lector Universitar.*

Departamentul Informatică şi IS,

Facultatea FCIM, UTM

**Chișinău 2024**

Cuprins:

[Scopul lucrării: 3](#_Toc164154077)

[Sarcina 3](#_Toc164154078)

[Concluzie: 17](#_Toc164154079)

Scopul lucrării:

Obținerea deprinderilor practice de implementare și de utilizare a tipului abstract de date „Listă simplu înlănțuită” în limbajul C cu asigurarea operațiilor de prelucrare de bază ale listei.

# Sarcina

Să se scrie 3 fișiere-text în limbajul C pentru implementarea și utilizarea tipului abstract de date „Listă simplu înlănțuită” cu asigurarea operațiilor de prelucrare de bază ale listei:

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

2) Fișier cu extensia .c, care conține implementările (codurile) funcțiilor declarate în fișierul antet.

3) Fișier al utilizatorului, funcția mаin() pentru prelucrarea listei cu afișarea la ecran a următorului meniu de opțiuni de bază:

1. Crearea listei în memoria dinamică

2. Introducerea informației despre elementele listei de la tastatură.

3. Afișarea 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. Interschimbarea a două elemente indicate în listă.

9. Sortarea listei.

10. Eliberarea memoriei alocate pentru listă.

0. Ieșire din program.

**Variante pentru structuri:**

5. Structura Restaurant cu câmpurile: denumirea, adresa, telefonul, categoria, numărul de locuri, specializarea.

**Codul:**

**#include <stdio.h>**

**#include <stdlib.h>**

**#include <string.h>**

**#define MAX\_NAME\_LENGTH 50**

**#define MAX\_ADDRESS\_LENGTH 100**

**#define MAX\_PHONE\_LENGTH 15**

**#define MAX\_CATEGORY\_LENGTH 20**

**#define MAX\_SPECIALIZATION\_LENGTH 50**

**struct Restaurant {**

**char denumirea[MAX\_NAME\_LENGTH];**

**char adresa[MAX\_ADDRESS\_LENGTH];**

**char telefonul[MAX\_PHONE\_LENGTH];**

**char categoria[MAX\_CATEGORY\_LENGTH];**

**int numar\_locuri;**

**char specializarea[MAX\_SPECIALIZATION\_LENGTH];**

**};**

**struct Restaurant \*lista\_restaurante = NULL;**

**int lungime\_lista = 0;**

**void salvare\_in\_fisier(const char \*nume\_fisier);**

**void incarcare\_din\_fisier(const char \*nume\_fisier);**

**void adauga\_restaurant();**

**void afiseaza\_restaurante();**

**int cauta\_restaurant(const char \*denumire);**

**void modifica\_restaurant();**

**void adauga\_locuri();**

**void adauga\_specializare();**

**void adauga\_telefon();**

**void interschimba\_restaurante();**

**void sorteaza\_restaurante();**

**void elibereaza\_memorie();**

**int main() {**

**incarcare\_din\_fisier("restaurante.txt");**

**int optiune;**

**do {**

**printf("\nMeniu:\n");**

**printf("1. Adauga restaurant\n");**

**printf("2. Afiseaza restaurantele\n");**

**printf("3. Cauta restaurant\n");**

**printf("4. Modifica restaurant\n");**

**printf("5. Adauga locuri\n");**

**printf("6. Adauga specializare\n");**

**printf("7. Adauga telefon\n");**

**printf("8. Interschimba restaurante\n");**

**printf("9. Sorteaza restaurantele\n");**

**printf("10. Elibereaza memoria\n");**

**printf("0. Iesire\n");**

**printf("Alegeti o optiune: ");**

**scanf("%d", &optiune);**

**switch(optiune) {**

**case 1:**

**adauga\_restaurant();**

**break;**

**case 2:**

**afiseaza\_restaurante();**

**break;**

**case 3: {**

**char denumire\_cautata[MAX\_NAME\_LENGTH];**

**printf("Introduceti denumirea restaurantului de cautat: ");**

**scanf(" %[^\n]", denumire\_cautata);**

**int pozitie = cauta\_restaurant(denumire\_cautata);**

**if (pozitie != -1) {**

**printf("Restaurantul a fost gasit la pozitia %d.\n", pozitie + 1);**

**} else {**

**printf("Restaurantul nu a fost gasit.\n");**

**}**

**break;**

**}**

**case 4:**

**modifica\_restaurant();**

**break;**

**case 5:**

**adauga\_locuri();**

**break;**

**case 6:**

**adauga\_specializare();**

**break;**

**case 7:**

**adauga\_telefon();**

**break;**

**case 8:**

**interschimba\_restaurante();**

**break;**

**case 9:**

**sorteaza\_restaurante();**

**break;**

**case 10:**

**elibereaza\_memorie();**

**break;**

**case 0:**

**printf("Iesire din program.\n");**

**break;**

**default:**

**printf("Optiune invalida. Va rugam sa alegeti o optiune valida.\n");**

**}**

**} while (optiune != 0);**

**salvare\_in\_fisier("restaurante.txt");**

**return 0;**

**}**

**void salvare\_in\_fisier(const char \*nume\_fisier) {**

**FILE \*f = fopen(nume\_fisier, "w");**

**if (f == NULL) {**

**printf("Eroare la deschiderea fisierului pentru scriere.\n");**

**return;**

**}**

**for (int i = 0; i < lungime\_lista; i++) {**

**fprintf(f, "%s|%s|%s|%s|%d|%s\n", lista\_restaurante[i].denumirea, lista\_restaurante[i].adresa, lista\_restaurante[i].telefonul, lista\_restaurante[i].categoria, lista\_restaurante[i].numar\_locuri, lista\_restaurante[i].specializarea);**

**}**

**fclose(f);**

**}**

**void incarcare\_din\_fisier(const char \*nume\_fisier) {**

**FILE \*f = fopen(nume\_fisier, "r");**

**if (f == NULL) {**

**printf("Eroare la deschiderea fisierului pentru citire.\n");**

**return;**

**}**

**char linie[1000];**

**while (fgets(linie, sizeof(linie), f)) {**

**struct Restaurant restaurant\_nou;**

**sscanf(linie, "%[^|]|%[^|]|%[^|]|%[^|]|%d|%[^\n]", restaurant\_nou.denumirea, restaurant\_nou.adresa, restaurant\_nou.telefonul, restaurant\_nou.categoria, &restaurant\_nou.numar\_locuri, restaurant\_nou.specializarea);**

**lista\_restaurante = (struct Restaurant \*)realloc(lista\_restaurante, (lungime\_lista + 1) \* sizeof(struct Restaurant));**

**if (lista\_restaurante == NULL) {**

**printf("Eroare la realocarea memoriei.\n");**

**fclose(f);**

**return;**

**}**

**lista\_restaurante[lungime\_lista] = restaurant\_nou;**

**lungime\_lista++;**

**}**

**fclose(f);**

**}**

**void adauga\_restaurant() {**

**struct Restaurant restaurant\_nou;**

**printf("Introduceti denumirea restaurantului: ");**

**scanf(" %[^\n]", restaurant\_nou.denumirea);**

**printf("Introduceti adresa restaurantului: ");**

**scanf(" %[^\n]", restaurant\_nou.adresa);**

**printf("Introduceti telefonul restaurantului: ");**

**scanf(" %[^\n]", restaurant\_nou.telefonul);**

**printf("Introduceti categoria restaurantului: ");**

**scanf(" %[^\n]", restaurant\_nou.categoria);**

**printf("Introduceti numarul de locuri al restaurantului: ");**

**scanf("%d", &restaurant\_nou.numar\_locuri);**

**printf("Introduceti specializarea restaurantului: ");**

**scanf(" %[^\n]", restaurant\_nou.specializarea);**

**lista\_restaurante = (struct Restaurant \*)realloc(lista\_restaurante, (lungime\_lista + 1) \* sizeof(struct Restaurant));**

**if (lista\_restaurante == NULL) {**

**printf("Eroare la realocarea memoriei.\n");**

**return;**

**}**

**lista\_restaurante[lungime\_lista] = restaurant\_nou;**

**lungime\_lista++;**

**printf("Restaurantul a fost adaugat cu succes.\n");**

**}**

**void afiseaza\_restaurante() {**

**if (lungime\_lista == 0) {**

**printf("Nu exista restaurante de afisat.\n");**

**return;**

**}**

**printf("Lista restaurantelor:\n");**

**for (int i = 0; i < lungime\_lista; i++) {**

**printf("Restaurant %d:\n", i + 1);**

**printf("Denumire: %s\n", lista\_restaurante[i].denumirea);**

**printf("Adresa: %s\n", lista\_restaurante[i].adresa);**

**printf("Telefon: %s\n", lista\_restaurante[i].telefonul);**

**printf("Categoria: %s\n", lista\_restaurante[i].categoria);**

**printf("Numar locuri: %d\n", lista\_restaurante[i].numar\_locuri);**

**printf("Specializare: %s\n", lista\_restaurante[i].specializarea);**

**}**

**}**

**int cauta\_restaurant(const char \*denumire) {**

**for (int i = 0; i < lungime\_lista; i++) {**

**if (strcmp(lista\_restaurante[i].denumirea, denumire) == 0) {**

**return i;**

**}**

**}**

**return -1;**

**}**

**void modifica\_restaurant() {**

**char denumire\_modificare[MAX\_NAME\_LENGTH];**

**printf("Introduceti denumirea restaurantului de modificat: ");**

**scanf(" %[^\n]", denumire\_modificare);**

**int pozitie = cauta\_restaurant(denumire\_modificare);**

**if (pozitie != -1) {**

**printf("Introduceti noile informatii pentru restaurant:\n");**

**printf("Denumire: ");**

**scanf(" %[^\n]", lista\_restaurante[pozitie].denumirea);**

**printf("Adresa: ");**

**scanf(" %[^\n]", lista\_restaurante[pozitie].adresa);**

**printf("Telefon: ");**

**scanf(" %[^\n]", lista\_restaurante[pozitie].telefonul);**

**printf("Categoria: ");**

**scanf(" %[^\n]", lista\_restaurante[pozitie].categoria);**

**printf("Numar locuri: ");**

**scanf("%d", &lista\_restaurante[pozitie].numar\_locuri);**

**printf("Specializare: ");**

**scanf(" %[^\n]", lista\_restaurante[pozitie].specializarea);**

**printf("Restaurantul a fost modificat cu succes.\n");**

**} else {**

**printf("Restaurantul nu a fost gasit.\n");**

**}**

**}**

**void adauga\_locuri() {**

**char denumire[MAX\_NAME\_LENGTH];**

**printf("Introduceti denumirea restaurantului: ");**

**scanf(" %[^\n]", denumire);**

**int pozitie = cauta\_restaurant(denumire);**

**if (pozitie != -1) {**

**int locuri\_suplimentare;**

**printf("Introduceti numarul de locuri suplimentare: ");**

**scanf("%d", &locuri\_suplimentare);**

**lista\_restaurante[pozitie].numar\_locuri += locuri\_suplimentare;**

**printf("Numarul de locuri a fost actualizat cu succes.\n");**

**} else {**

**printf("Restaurantul nu a fost gasit.\n");**

**}**

**}**

**void adauga\_specializare() {**

**char denumire[MAX\_NAME\_LENGTH];**

**printf("Introduceti denumirea restaurantului: ");**

**scanf(" %[^\n]", denumire);**

**int pozitie = cauta\_restaurant(denumire);**

**if (pozitie != -1) {**

**printf("Introduceti noua specializare: ");**

**scanf(" %[^\n]", lista\_restaurante[pozitie].specializarea);**

**printf("Specializarea a fost actualizata cu succes.\n");**

**} else {**

**printf("Restaurantul nu a fost gasit.\n");**

**}**

**}**

**void adauga\_telefon() {**

**char denumire[MAX\_NAME\_LENGTH];**

**printf("Introduceti denumirea restaurantului: ");**

**scanf(" %[^\n]", denumire);**

**int pozitie = cauta\_restaurant(denumire);**

**if (pozitie != -1) {**

**printf("Introduceti noul numar de telefon: ");**

**scanf(" %[^\n]", lista\_restaurante[pozitie].telefonul);**

**printf("Numarul de telefon a fost actualizat cu succes.\n");**

**} else {**

**printf("Restaurantul nu a fost gasit.\n");**

**}**

**}**

**void interschimba\_restaurante() {**

**int pozitie1, pozitie2;**

**printf("Introduceti pozitiile restaurantelor de interschimbat: ");**

**scanf("%d %d", &pozitie1, &pozitie2);**

**pozitie1--;**

**pozitie2--;**

**if (pozitie1 >= 0 && pozitie1 < lungime\_lista && pozitie2 >= 0 && pozitie2 < lungime\_lista) {**

**struct Restaurant temp = lista\_restaurante[pozitie1];**

**lista\_restaurante[pozitie1] = lista\_restaurante[pozitie2];**

**lista\_restaurante[pozitie2] = temp;**

**printf("Restaurantele au fost interschimbate cu succes.\n");**

**} else {**

**printf("Pozitii invalide.\n");**

**}**

**}**

**void sorteaza\_restaurante() {**

**for (int i = 0; i < lungime\_lista - 1; i++) {**

**for (int j = 0; j < lungime\_lista - i - 1; j++) {**

**if (strcmp(lista\_restaurante[j].denumirea, lista\_restaurante[j + 1].denumirea) > 0) {**

**struct Restaurant temp = lista\_restaurante[j];**

**lista\_restaurante[j] = lista\_restaurante[j + 1];**

**lista\_restaurante[j + 1] = temp;**

**}**

**}**

**}**

**printf("Restaurantele au fost sortate cu succes.\n");**

**}**

**void elibereaza\_memorie() {**

**free(lista\_restaurante);**

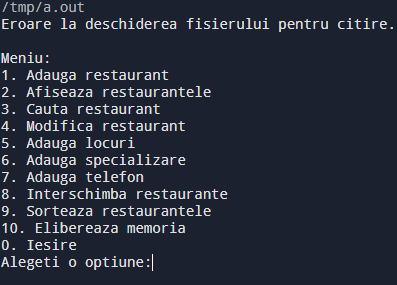
**lista\_restaurante = NULL;**

**lungime\_lista = 0;**

**printf("Memoria a fost eliberata.\n");**

**}**

**Rezultatele:**

****

# Concluzie:

Implementarea și utilizarea tipului abstract de date "Listă simplu înlănțuită" în limbajul C necesită organizare și atenție la detalii. Divizarea codului în fișiere separate pentru antet, implementare și utilizator permite o gestionare eficientă a codului și separă interfața publică de detaliile de implementare. Prin respectarea acestor principii, se obține un program robust și ușor de întreținut, care oferă funcționalitatea necesară pentru manipularea listelor simplu înlănțuite.