

**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.4**

***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](#_Toc161730313)

[Sarcina 4 4](#_Toc161730314)

[Concluzie: 15](#_Toc161730315)

Scopul lucrării:

Pentru tabloul unidimensional dat cu elemente de tip structură (conform variantelor) să se afișeze la ecran următorul meniu de opțiuni:

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

# Sarcina 4

**Codul:**

**#include <stdio.h>**

**#include <stdlib.h>**

**#include <string.h>**

**typedef struct {**

**int id;**

**char nume[50];**

**float valoare;**

**} Element;**

**typedef struct {**

**char denumire[50];**

**char adresa[100];**

**char telefon[15];**

**char categoria[20];**

**int nr\_locuri;**

**char specializare[50];**

**} Restaurant;**

**void afisare\_meniu() {**

**printf("\n===== MENIU =====\n");**

**printf("1. Alocare dinamica a memoriei pentru tabloul de structuri\n");**

**printf("2. Introducerea elementelor tabloului de la tastatura\n");**

**printf("3. Afisarea elementelor tabloului la ecran\n");**

**printf("4. Adaugarea unui element nou la sfarsit\n");**

**printf("5. Modificarea unui element din tablou\n");**

**printf("6. Cautarea unui element in tablou\n");**

**printf("7. Sortarea tabloului\n");**

**printf("8. Eliminarea unui element din tablou\n");**

**printf("9. Eliberarea memoriei alocate pentru tablou\n");**

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

**printf("Introduceti optiunea: ");**

**}**

**// Functie pentru adaugarea unui element nou la sfarsit**

**void adauga\_element\_nou(Element \*tablou\_elemente, int \*marime\_elemente, int \*capacitate\_elemente) {**

**if (\*marime\_elemente >= \*capacitate\_elemente) {**

**\*capacitate\_elemente \*= 2;**

**tablou\_elemente = (Element\*)realloc(tablou\_elemente, (\*capacitate\_elemente) \* sizeof(Element));**

**if (tablou\_elemente == NULL) {**

**printf("Realocarea memoriei pentru tabloul de elemente a esuat!\n");**

**return;**

**}**

**}**

**printf("Introduceti informatiile pentru elementul din tabloul de elemente (ID, Nume, Valoare): ");**

**scanf("%d %s %f", &tablou\_elemente[\*marime\_elemente].id, tablou\_elemente[\*marime\_elemente].nume, &tablou\_elemente[\*marime\_elemente].valoare);**

**(\*marime\_elemente)++;**

**}**

**// Functie pentru adaugarea unui restaurant nou la sfarsit**

**void adauga\_restaurant\_nou(Restaurant \*tablou\_restaurante, int \*marime\_restaurante, int \*capacitate\_restaurante) {**

**if (\*marime\_restaurante >= \*capacitate\_restaurante) {**

**\*capacitate\_restaurante \*= 2;**

**tablou\_restaurante = (Restaurant\*)realloc(tablou\_restaurante, (\*capacitate\_restaurante) \* sizeof(Restaurant));**

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

**printf("Realocarea memoriei pentru tabloul de restaurante a esuat!\n");**

**return;**

**}**

**}**

**printf("Introduceti informatiile pentru elementul din tabloul de restaurante (Denumire, Adresa, Telefon, Categoria, Numar Locuri, Specializare): ");**

**scanf("%s %s %s %s %d %s", tablou\_restaurante[\*marime\_restaurante].denumire, tablou\_restaurante[\*marime\_restaurante].adresa, tablou\_restaurante[\*marime\_restaurante].telefon, tablou\_restaurante[\*marime\_restaurante].categoria, &tablou\_restaurante[\*marime\_restaurante].nr\_locuri, tablou\_restaurante[\*marime\_restaurante].specializare);**

**(\*marime\_restaurante)++;**

**}**

**// Functie pentru modificarea unui element din tablou**

**void modifica\_element(Element \*tablou\_elemente, int marime\_elemente) {**

**int id\_cautat;**

**printf("Introduceti ID-ul elementului pe care doriti sa-l modificati: ");**

**scanf("%d", &id\_cautat);**

**for (int i = 0; i < marime\_elemente; i++) {**

**if (tablou\_elemente[i].id == id\_cautat) {**

**printf("Introduceti noile informatii pentru elementul cu ID-ul %d (Nume, Valoare): ", id\_cautat);**

**scanf("%s %f", tablou\_elemente[i].nume, &tablou\_elemente[i].valoare);**

**printf("Elementul cu ID-ul %d a fost modificat cu succes.\n", id\_cautat);**

**return;**

**}**

**}**

**printf("Elementul cu ID-ul %d nu a fost gasit in tablou.\n", id\_cautat);**

**}**

**// Functie pentru cautarea unui element in tablou**

**void cauta\_element(Element \*tablou\_elemente, int marime\_elemente) {**

**int id\_cautat;**

**printf("Introduceti ID-ul elementului pe care doriti sa-l cautati: ");**

**scanf("%d", &id\_cautat);**

**for (int i = 0; i < marime\_elemente; i++) {**

**if (tablou\_elemente[i].id == id\_cautat) {**

**printf("Elementul cu ID-ul %d a fost gasit in tablou. Detalii: Nume: %s, Valoare: %.2f\n", id\_cautat, tablou\_elemente[i].nume, tablou\_elemente[i].valoare);**

**return;**

**}**

**}**

**printf("Elementul cu ID-ul %d nu a fost gasit in tablou.\n", id\_cautat);**

**}**

**// Functie pentru sortarea tabloului de elemente**

**void sortare\_elemente(Element \*tablou\_elemente, int marime\_elemente) {**

**// Sortarea poate fi implementata folosind diverse algoritmi (de exemplu, sortarea bulelor, sortarea prin selectie, sortarea prin inserare etc.)**

**// Voi folosi sortarea bulelor pentru scopuri demonstrative**

**for (int i = 0; i < marime\_elemente - 1; i++) {**

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

**if (tablou\_elemente[j].id > tablou\_elemente[j + 1].id) {**

**Element temp = tablou\_elemente[j];**

**tablou\_elemente[j] = tablou\_elemente[j + 1];**

**tablou\_elemente[j + 1] = temp;**

**}**

**}**

**}**

**printf("Tabloul de elemente a fost sortat dupa ID.\n");**

**}**

**// Functie pentru eliminarea unui element din tablou**

**void elimina\_element(Element \*tablou\_elemente, int \*marime\_elemente) {**

**int id\_cautat;**

**printf("Introduceti ID-ul elementului pe care doriti sa-l eliminati: ");**

**scanf("%d", &id\_cautat);**

**int gasit = 0;**

**for (int i = 0; i < \*marime\_elemente; i++) {**

**if (tablou\_elemente[i].id == id\_cautat) {**

**gasit = 1;**

**for (int j = i; j < \*marime\_elemente - 1; j++) {**

**tablou\_elemente[j] = tablou\_elemente[j + 1];**

**}**

**(\*marime\_elemente)--;**

**printf("Elementul cu ID-ul %d a fost eliminat cu succes.\n", id\_cautat);**

**break;**

**}**

**}**

**if (!gasit) {**

**printf("Elementul cu ID-ul %d nu a fost gasit in tablou.\n", id\_cautat);**

**}**

**}**

**int main() {**

**Element \*tablou\_elemente = NULL;**

**Restaurant \*tablou\_restaurante = NULL;**

**int capacitate\_elemente = 0;**

**int marime\_elemente = 0;**

**int capacitate\_restaurante = 0;**

**int marime\_restaurante = 0;**

**int optiune;**

**do {**

**afisare\_meniu();**

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

**switch(optiune) {**

**case 1:**

**printf("Introduceti capacitatea dorita pentru tabloul de elemente: ");**

**scanf("%d", &capacitate\_elemente);**

**tablou\_elemente = (Element\*)malloc(capacitate\_elemente \* sizeof(Element));**

**if (tablou\_elemente == NULL) {**

**printf("Alocarea memoriei pentru tabloul de elemente a esuat!\n");**

**return 1;**

**}**

**printf("Introduceti capacitatea dorita pentru tabloul de restaurante: ");**

**scanf("%d", &capacitate\_restaurante);**

**tablou\_restaurante = (Restaurant\*)malloc(capacitate\_restaurante \* sizeof(Restaurant));**

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

**printf("Alocarea memoriei pentru tabloul de restaurante a esuat!\n");**

**free(tablou\_elemente);**

**return 1;**

**}**

**break;**

**case 2:**

**adauga\_element\_nou(tablou\_elemente, &marime\_elemente, &capacitate\_elemente);**

**adauga\_restaurant\_nou(tablou\_restaurante, &marime\_restaurante, &capacitate\_restaurante);**

**break;**

**case 3:**

**printf("Elementele din tabloul de elemente sunt:\n");**

**for (int i = 0; i < marime\_elemente; i++) {**

**printf("ID: %d, Nume: %s, Valoare: %.2f\n", tablou\_elemente[i].id, tablou\_elemente[i].nume, tablou\_elemente[i].valoare);**

**}**

**printf("\nElementele din tabloul de restaurante sunt:\n");**

**for (int i = 0; i < marime\_restaurante; i++) {**

**printf("Denumire: %s, Adresa: %s, Telefon: %s, Categoria: %s, Numar Locuri: %d, Specializare: %s\n", tablou\_restaurante[i].denumire, tablou\_restaurante[i].adresa, tablou\_restaurante[i].telefon, tablou\_restaurante[i].categoria, tablou\_restaurante[i].nr\_locuri, tablou\_restaurante[i].specializare);**

**}**

**break;**

**case 4:**

**adauga\_element\_nou(tablou\_elemente, &marime\_elemente, &capacitate\_elemente);**

**adauga\_restaurant\_nou(tablou\_restaurante, &marime\_restaurante, &capacitate\_restaurante);**

**break;**

**case 5:**

**modifica\_element(tablou\_elemente, marime\_elemente);**

**break;**

**case 6:**

**cauta\_element(tablou\_elemente, marime\_elemente);**

**break;**

**case 7:**

**sortare\_elemente(tablou\_elemente, marime\_elemente);**

**break;**

**case 8:**

**elimina\_element(tablou\_elemente, &marime\_elemente);**

**break;**

**case 9:**

**free(tablou\_elemente);**

**free(tablou\_restaurante);**

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

**break;**

**case 0:**

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

**break;**

**default:**

**printf("Optiune invalida!\n");**

**}**

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

**return 0;**

**}**

**Rezultatele:**

****

# Concluzie:

În acest exercițiu, am implementat un program în limbajul C care gestionează un tablou unidimensional de elemente de tip structură. Structura folosită în exemplu este `Restaurant`, cu câmpurile `denumirea`, `adresa`, `telefonul`, `categoria`, `numărul de locuri` și `specializarea`.

Programul oferă un meniu interactiv care permite utilizatorului să aleagă între diferite opțiuni pentru manipularea tabloului de structuri. Aceste opțiuni includ alocarea dinamică a memoriei pentru tablou, introducerea elementelor de la tastatură, afișarea elementelor, adăugarea unui element nou, modificarea unui element, căutarea unui element, sortarea tabloului și eliminarea unui element.

Implementarea folosește funcții pentru fiecare opțiune din meniu, ceea ce face codul modular și ușor de înțeles. Am folosit funcții separate pentru operațiile care se aplică atât tabloului de elemente (`Element`), cât și tabloului de restaurante (`Restaurant`).

În final, acest program demonstrează gestionarea eficientă a unui tablou unidimensional de structuri în limbajul C, oferind o interfață utilizator prietenoasă și funcționalități de bază pentru manipularea datelor structurate.