

**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.5**

***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](#_Toc162267062)

[Sarcina 1 4](#_Toc162267063)

[Concluzie: 4](#_Toc162267064)

Scopul lucrării:

Programarea algoritmilor de prelucrare a structurilor și a tablourilor de tip structură prin utilizarea funcțiilor, pointerilor, alocării dinamice a memoriei și manipularea datelor dintr-un fișier textual și binar.

# Sarcina 1

1. Introducerea elementelor tabloului de la tastatură.

2. Afișarea elementelor tabloului într-un fișier textual.

3. Afișarea elementelor tabloului într-un fișier binar.

4. Sortarea în ordinea inversă lexicografiei a datelor din tablou după primul câmp de tip string, utilizând o tehica de sotare la alegere și înscrierea datelor sortate într-un fișier textual.

5. Sortarea în creștere a datelor din tablou după un câmp de tip întreg, utilizând o tehica de sotare la alegere și înscrierea datelor sortate într-un fișier binar.

6. Afișarea în consolă a conținutului fișierului textual creat cu datele sortate.

7. Afișarea în consolă a conținutului fișierului binar creat cu datele sortate.

Structura Capitală cu câmpurile: denumirea, țară, populația, suprafața, primarul.

**Codul:  
#include <stdio.h>**

**#include <stdlib.h>**

**#include <string.h>**

**// Definirea structurii Capitală**

**typedef struct {**

**char \*denumirea;**

**char \*țară;**

**int populație;**

**float suprafața;**

**char \*primarul;**

**} Capitală;**

**// Funcție pentru citirea datelor de la tastatură și alocarea dinamică a memoriei pentru structură**

**void citire\_date(Capitală \*capitale, int n) {**

**for (int i = 0; i < n; i++) {**

**capitale[i].denumirea = (char \*)malloc(100 \* sizeof(char));**

**capitale[i].țară = (char \*)malloc(100 \* sizeof(char));**

**capitale[i].primarul = (char \*)malloc(100 \* sizeof(char));**

**printf("Introduceți denumirea capitalei %d: ", i + 1);**

**scanf("%s", capitale[i].denumirea);**

**printf("Introduceți țara pentru capitala %s: ", capitale[i].denumirea);**

**scanf("%s", capitale[i].țară);**

**printf("Introduceți populația pentru capitala %s: ", capitale[i].denumirea);**

**scanf("%d", &capitale[i].populație);**

**printf("Introduceți suprafața pentru capitala %s: ", capitale[i].denumirea);**

**scanf("%f", &capitale[i].suprafața);**

**printf("Introduceți numele primarului pentru capitala %s: ", capitale[i].denumirea);**

**scanf("%s", capitale[i].primarul);**

**}**

**}**

**// Funcție pentru afișarea datelor din tablou într-un fișier textual**

**void afișare\_textuală(Capitală \*capitale, int n, const char \*nume\_fișier) {**

**FILE \*fișier = fopen(nume\_fișier, "w");**

**if (fișier == NULL) {**

**printf("Eroare la deschiderea fișierului!\n");**

**exit(EXIT\_FAILURE);**

**}**

**for (int i = 0; i < n; i++) {**

**fprintf(fișier, "%s %s %d %.2f %s\n", capitale[i].denumirea, capitale[i].țară, capitale[i].populație,**

**capitale[i].suprafața, capitale[i].primarul);**

**}**

**fclose(fișier);**

**}**

**// Funcție pentru afișarea datelor din tablou într-un fișier binar**

**void afișare\_binară(Capitală \*capitale, int n, const char \*nume\_fișier) {**

**FILE \*fișier = fopen(nume\_fișier, "wb");**

**if (fișier == NULL) {**

**printf("Eroare la deschiderea fișierului!\n");**

**exit(EXIT\_FAILURE);**

**}**

**for (int i = 0; i < n; i++) {**

**fwrite(&capitale[i], sizeof(Capitală), 1, fișier);**

**}**

**fclose(fișier);**

**}**

**// Funcție de comparare pentru sortarea lexicografică inversă după denumirea capitală**

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

**return strcmp(((Capitală \*)b)->denumirea, ((Capitală \*)a)->denumirea);**

**}**

**// Funcție de comparare pentru sortarea crescătoare după populație**

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

**return (((Capitală \*)a)->populație - ((Capitală \*)b)->populație);**

**}**

**// Funcție pentru afișarea conținutului unui fișier text**

**void afișare\_conținut\_textual(const char \*nume\_fișier) {**

**FILE \*fișier = fopen(nume\_fișier, "r");**

**if (fișier == NULL) {**

**printf("Eroare la deschiderea fișierului!\n");**

**exit(EXIT\_FAILURE);**

**}**

**char buffer[100];**

**while (fgets(buffer, sizeof(buffer), fișier) != NULL) {**

**printf("%s", buffer);**

**}**

**fclose(fișier);**

**}**

**// Funcție pentru afișarea conținutului unui fișier binar**

**void afișare\_conținut\_binar(const char \*nume\_fișier) {**

**FILE \*fișier = fopen(nume\_fișier, "rb");**

**if (fișier == NULL) {**

**printf("Eroare la deschiderea fișierului!\n");**

**exit(EXIT\_FAILURE);**

**}**

**Capitală capitală;**

**while (fread(&capitală, sizeof(Capitală), 1, fișier)) {**

**printf("%s %s %d %.2f %s\n", capitală.denumirea, capitală.țară, capitală.populație, capitală.suprafața,**

**capitală.primarul);**

**}**

**fclose(fișier);**

**}**

**int main() {**

**int n;**

**printf("Introduceți numărul de capitale: ");**

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

**// Alocare dinamică pentru tabloul de capitale**

**Capitală \*capitale = (Capitală \*)malloc(n \* sizeof(Capitală));**

**if (capitale == NULL) {**

**printf("Memorie insuficientă!\n");**

**return EXIT\_FAILURE;**

**}**

**// Citirea datelor de la tastatură**

**citire\_date(capitale, n);**

**// Afișarea datelor într-un fișier textual**

**afișare\_textuală(capitale, n, "capitale.txt");**

**// Afișarea datelor într-un fișier binar**

**afișare\_binară(capitale, n, "capitale.bin");**

**// Sortarea lexicografică inversă a datelor și afișarea într-un fișier text**

**qsort(capitale, n, sizeof(Capitală), comparare\_lexicografică\_inversă);**

**afișare\_textuală(capitale, n, "capitale\_sortate.txt");**

**// Sortarea crescătoare a datelor după populație și afișarea într-un fișier binar**

**qsort(capitale, n, sizeof(Capitală), comparare\_crescătoare);**

**afișare\_binară(capitale, n, "capitale\_sortate.bin");**

**// Afișarea conținutului fișierului textual cu datele sortate**

**printf("Conținutul fișierului textual cu datele sortate:\n");**

**afișare\_conținut\_textual("capitale\_sortate.txt");**

**// Afișarea conținutului fișierului binar cu datele sortate**

**printf("\nConținutul fișierului binar cu datele sortate:\n");**

**afișare\_conținut\_binar("capitale\_sortate.bin");**

**// Eliberarea memoriei alocate pentru structuri**

**for (int i = 0; i < n; i++) {**

**free(capitale[i].denumirea);**

**free(capitale[i].țară);**

**free(capitale[i].primarul);**

**}**

**free(capitale);**

**return 0;**

**}**

**Rezultatele:**

# Concluzie:

Această lucrare a prezentat un program în limbajul C care efectuează mai multe operații cu un tablou de structuri, denumite "Capitală". Programul începe prin citirea datelor despre capitale de la tastatură și le stochează într-un tablou de structuri alocat dinamic. Apoi, datele sunt afișate atât într-un fișier text, cât și într-un fișier binar.

Următoarea parte a programului implică sortarea datelor. În primul rând, datele sunt sortate în ordine inversă lexicografică după primul câmp de tip string (denumirea capitalei), iar apoi sunt sortate în ordine crescătoare după populație. Sortarea se realizează utilizând funcția `qsort` din biblioteca standard a limbajului C.

După sortare, datele sunt înscrise în fișierele corespunzătoare și afișate în consolă, atât pentru fișierul text cât și pentru fișierul binar.

Concluzionând, acest program demonstrează utilizarea conceptelor fundamentale ale limbajului C pentru manipularea și prelucrarea datelor într-un mediu structurat. Acesta acoperă aspecte precum alocarea dinamică a memoriei, citirea și scrierea în fișiere, sortarea și afișarea datelor, toate fiind operații comune în dezvoltarea de aplicații software. De asemenea, utilizarea funcțiilor și a pointerilor este evidențiată pentru a gestiona eficient structurile de date și memoria asociată acestora.