****

**MINISTERUL EDUCAȚIEI, CULTURII ȘI CERCETĂRII**

**AL REPUBLICII MOLDOVA**

**Universitatea Tehnică a Moldovei**

**Facultatea Calculatoare, Informatică şi Microelectronică**

**Departamentul Informatică şi Ingineria Sistemelor**

**Calancea Cătălin**

**MI-222**

**Raport**

**pentru lucrarea de laborator Nr.6**

***la cursul de “Structuri de date şi algoritmi”***

Verificat:

**Guțu Maria,** *doctor, conf. univ.*

Departamentul Informatică şi IS,

Facultatea FCIM, UTM

**Chișinău – 2023**

**Scopul lucrării:** Programarea algoritmilor de prelucrare a fișierelor binare prin utilizarea structurilor, funcțiilor, pointerilor, alocării dinamice a memoriei în limbajul C.

**Nr. variantei:** 4

**Condiția problemei ( sarcinii de lucru ) :**

Pentru fișierele binare cu elemente de tip structură (conform variantelor) să se afișeze la ecran următorul meniu de opțiuni:

1. Crearea unui fișier binar nou, introducerea câmpurilor structurilor de la tastatură și înscrierea lor în acest fișier.   
2. Afișarea elementelor fișierului binar la ecran.  
3. Adăugarea unei structuri noi la sfârșit de fișier.  
4. Modificarea unei structuri a fișierului.  
5. Căutarea după un câmp al structurii în fișier.  
6. Sortarea structurilor fișierului după un careva câmp.   
7. Eliminarea unei structuri din fișier.  
8. Ștergerea fișierului de pe disc (la dorință)  
0. Ieșire din program.   
  
Structura Cinematograf cu câmpurile: denumirea, adresa, telefonul, numărul de locuri, prețul biletelor.

**Codul programului:**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#define MAX\_CINEMAS 100

struct Cinema {

char name[50];

char address[100];

char phone[15];

int seats;

float ticket\_price;

};

// function prototypes

void createFile();

void displayFile();

void addRecord();

void modifyRecord();

void searchRecord();

void sortRecords();

void deleteRecord();

void deleteFile();

int main() {

int choice;

do {

printf("\n\nCinema Record Management System");

printf("\n1. Create new file");

printf("\n2. Display file contents");

printf("\n3. Add new record to file");

printf("\n4. Modify record in file");

printf("\n5. Search for record in file");

printf("\n6. Sort records in file");

printf("\n7. Delete record from file");

printf("\n8. Delete file");

printf("\n0. Exit program");

printf("\nEnter your choice: ");

scanf("%d", &choice);

switch(choice) {

case 1:

createFile();

break;

case 2:

displayFile();

break;

case 3:

addRecord();

break;

case 4:

modifyRecord();

break;

case 5:

searchRecord();

break;

case 6:

sortRecords();

break;

case 7:

deleteRecord();

break;

case 8:

deleteFile();

break;

case 0:

printf("\nExiting program...");

break;

default:

printf("\nInvalid choice. Try again.");

}

} while(choice != 0);

return 0;

}

void createFile() {

struct Cinema cinema;

FILE \*fp;

fp = fopen("cinemas.bin", "wb");

if(fp == NULL) {

printf("\nFailed to create file.");

return;

}

printf("\nEnter details of cinemas.");

printf("\nEnter '0' to stop adding records.\n");

while(1) {

printf("\nName: ");

fflush(stdin);

gets(cinema.name);

if(strcmp(cinema.name, "0") == 0) {

break;

}

printf("Address: ");

fflush(stdin);

gets(cinema.address);

printf("Phone: ");

fflush(stdin);

gets(cinema.phone);

printf("Seats: ");

scanf("%d", &cinema.seats);

printf("Ticket price: ");

scanf("%f", &cinema.ticket\_price);

fwrite(&cinema, sizeof(struct Cinema), 1, fp);

}

fclose(fp);

printf("\nFile created successfully.");

}

void displayFile() {

struct Cinema cinema;

FILE \*fp;

fp = fopen("cinemas.bin", "rb");

if(fp == NULL) {

printf("\nFile not found.");

return;

}

printf("\n%-20s %-30s %-15s %-10s %-10s", "Name", "Address", "Phone", "Seats", "Ticket Price");

while(fread(&cinema, sizeof(struct Cinema), 1, fp) == 1) {

printf("\n%-20s %-30s %-15s %-10d %-10.2f", cinema.name, cinema.address, cinema.phone, cinema.seats, cinema.ticket\_price);

}

fclose(fp);

}

void addRecord() {

struct Cinema cinema;

FILE \*fp;

fp = fopen("cinemas.bin", "ab");

if(fp == NULL) {

printf("\nFile not found.");

return;

}

printf("\nEnter details of new cinema.");

printf("\nName: ");

fflush(stdin);

gets(cinema.name);

printf("Address: ");

fflush(stdin);

gets(cinema.address);

printf("Phone: ");

fflush(stdin);

gets(cinema.phone);

printf("Seats: ");

scanf("%d", &cinema.seats);

printf("Ticket price: ");

scanf("%f", &cinema.ticket\_price);

fwrite(&cinema, sizeof(struct Cinema), 1, fp);

fclose(fp);

printf("\nRecord added successfully.");

}

void modifyRecord() {

struct Cinema cinema;

char name[50];

int found = 0;

FILE \*fp, \*temp;

fp = fopen("cinemas.bin", "rb");

if(fp == NULL) {

printf("\nFile not found.");

return;

}

temp = fopen("temp.bin", "wb");

if(temp == NULL) {

printf("\nFailed to create temporary file.");

fclose(fp);

return;

}

printf("\nEnter name of cinema to modify: ");

fflush(stdin);

gets(name);

while(fread(&cinema, sizeof(struct Cinema), 1, fp) == 1) {

if(strcmp(name, cinema.name) == 0) {

printf("\nCurrent details:");

printf("\nName: %s", cinema.name);

printf("\nAddress: %s", cinema.address);

printf("\nPhone: %s", cinema.phone);

printf("\nSeats: %d", cinema.seats);

printf("\nTicket price: %.2f", cinema.ticket\_price);

printf("\n\nEnter new details:");

printf("\nName: ");

fflush(stdin);

gets(cinema.name);

printf("Address: ");

fflush(stdin);

gets(cinema.address);

printf("Phone: ");

fflush(stdin);

gets(cinema.phone);

printf("Seats: ");

scanf("%d", &cinema.seats);

printf("Ticket price: ");

scanf("%f", &cinema.ticket\_price);

fwrite(&cinema, sizeof(struct Cinema), 1, temp);

found = 1;

} else {

fwrite(&cinema, sizeof(struct Cinema), 1, temp);

}

}

fclose(fp);

fclose(temp);

if(found) {

remove("cinemas.bin");

rename("temp.bin", "cinemas.bin");

printf("\nRecord modified successfully.");

} else {

remove("temp.bin");

printf("\nRecord not found.");

}

}

void searchRecord() {

struct Cinema cinema;

char name[50];

int found = 0;

FILE \*fp;

fp = fopen("cinemas.bin", "rb");

if(fp == NULL) {

printf("\nFile not found.");

return;

}

printf("\nEnter name of cinema to search: ");

fflush(stdin);

gets(name);

while(fread(&cinema, sizeof(struct Cinema), 1, fp) == 1) {

if(strcmp(name, cinema.name) == 0) {

printf("\nDetails of cinema:");

printf("\nName: %s", cinema.name);

printf("\nAddress: %s", cinema.address);

printf("\nPhone: %s", cinema.phone);

printf("\nSeats: %d", cinema.seats);

printf("\nTicket price: %.2f", cinema.ticket\_price);

found = 1;

break;

}

}

fclose(fp);

if(!found) {

printf("\nRecord not found.");

}

}

void sortRecords() {

struct Cinema cinemas[50];

struct Cinema temp;

int count = 0, i, j, sort\_field;

FILE \*fp;

fp = fopen("cinemas.bin", "rb+");

if(fp == NULL) {

printf("\nFile not found.");

return;

}

while(fread(&cinemas[count], sizeof(struct Cinema), 1, fp) == 1) {

count++;

}

printf("\nSort records by:");

printf("\n1. Name");

printf("\n2. Address");

printf("\n3. Phone");

printf("\n4. Seats");

printf("\n5. Ticket price");

printf("\nEnter field number: ");

scanf("%d", &sort\_field);

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

for(j = i+1; j < count; j++) {

switch(sort\_field) {

case 1:

if(strcmp(cinemas[i].name, cinemas[j].name) > 0) {

temp = cinemas[i];

cinemas[i] = cinemas[j];

cinemas[j] = temp;

}

break;

case 2:

if(strcmp(cinemas[i].address, cinemas[j].address) > 0) {

temp = cinemas[i];

cinemas[i] = cinemas[j];

cinemas[j] = temp;

}

break;

case 3:

if(strcmp(cinemas[i].phone, cinemas[j].phone) > 0) {

temp = cinemas[i];

cinemas[i] = cinemas[j];

cinemas[j] = temp;

}

break;

case 4:

if(cinemas[i].seats > cinemas[j].seats) {

temp = cinemas[i];

cinemas[i] = cinemas[j];

cinemas[j] = temp;

}

break;

case 5:

if(cinemas[i].ticket\_price > cinemas[j].ticket\_price) {

temp = cinemas[i];

cinemas[i] = cinemas[j];

cinemas[j] = temp;

}

break;

default:

printf("\nInvalid field number.");

fclose(fp);

return;

}

}

}

rewind(fp);

for(i = 0; i < count; i++) {

fwrite(&cinemas[i], sizeof(struct Cinema), 1, fp);

}

fclose(fp);

printf("\nRecords sorted successfully.");

}

void deleteRecord() {

struct Cinema cinema;

char name[50];

int found = 0;

FILE \*fp, \*temp;

fp = fopen("cinemas.bin", "rb");

if(fp == NULL) {

printf("\nFile not found.");

return;

}

temp = fopen("temp.bin", "wb");

if(temp == NULL) {

printf("\nFailed to create temporary file.");

fclose(fp);

return;

}

printf("\nEnter name of cinema to delete: ");

fflush(stdin);

gets(name);

while(fread(&cinema, sizeof(struct Cinema), 1, fp) == 1) {

if(strcmp(name, cinema.name) == 0) {

printf("\nRecord deleted:");

printf("\nName: %s", cinema.name);

printf("\nAddress: %s", cinema.address);

printf("\nPhone: %s", cinema.phone);

printf("\nSeats: %d", cinema.seats);

printf("\nTicket price: %.2f", cinema.ticket\_price);

found = 1;

} else {

fwrite(&cinema, sizeof(struct Cinema), 1, temp);

}

}

fclose(fp);

fclose(temp);

if(found == 0) {

printf("\nRecord not found.");

remove("temp.bin");

} else {

remove("cinemas.bin");

rename("temp.bin", "cinemas.bin");

printf("\nRecord deleted successfully.");

}

}

void deleteFile() {

int choice;

printf("\nAre you sure you want to delete the file?");

printf("\n1. Yes");

printf("\n2. No");

printf("\nEnter your choice: ");

scanf("%d", &choice);

if(choice == 1) {

if(remove("cinemas.bin") == 0) {

printf("\nFile deleted successfully.");

} else {

printf("\nFailed to delete file.");

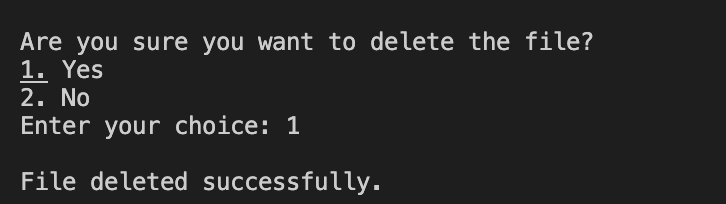
}

}

}

**Output:**

****

****

**În concluzie,** in cadrul acestei lucrări de laborator, am învățat cum să citim și să scriem fișiere binare utilizând diferite metode și am implementat o serie de funcții și structuri care ne-au ajutat să manipulăm datele din aceste fișiere. De asemenea, am învățat despre alocarea dinamică a memoriei și cum să o utilizăm în mod eficient pentru a gestiona datele în memoria sistemului.Această lucrare de laborator ne-a ajutat să înțelegem mai bine conceptele cheie ale programării în limbajul C și cum să le aplicăm pentru a prelucra fișiere binare într-un mod eficient și robust. De asemenea, am avut ocazia să exersăm gândirea logică și dezvoltarea de algoritmi pentru a rezolva probleme practice de programare.În general, această lucrare de laborator a fost un pas important în dezvoltarea abilităților noastre de programare și ne-a oferit o bază solidă pentru a explora în continuare programarea în limbajul C și dezvoltarea de aplicații software complexe.