****

**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.4**

***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 structurilor și a tablourilor de la structuri prin utilizarea funcțiilor, pointerilor, alocării dinamice a memoriei în limbajul C.

**Nr. variantei:** 4

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

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

**1. Introducerea elementelor tabloului de la tastatură.**

**2. Afișarea elementelor tabloului la ecran.**

**3. Sortarea tabloului crescător/descrescător(2 tehnici de sortare la alegere).**

**0. Ieșire din program.**

**Structura Cinematograf cu câmpurile: denumirea, adresa, telefonul, numărul de locuri, prețul biletelor.**

**Codul programului:**

#include <stdio.h>

#include <string.h>

#include <stdbool.h>

struct Cinematograf {

char denumirea[50];

char adresa[50];

char telefonul[20];

char nrlocuri[50];

int pret;

};

void read(struct Cinematograf \*st, int n);

void write(struct Cinematograf \*st, int n);

void bubble1(struct Cinematograf \*st, int n);

void bubble2(struct Cinematograf \*st, int n);

void insertion1(struct Cinematograf \*st, int n);

void insertion2(struct Cinematograf \*st, int n);

void clearBuffer();

void menu();

int main() {

int n, choice;

printf("The number of cinemas : ");

scanf("%d", &n);

clearBuffer();

struct Cinematograf st[10];

read(st, n);

do {

menu();

scanf("%d", &choice);

clearBuffer();

switch (choice) {

case 1:

read(st, n);

break;

case 2:

printf("\nDisplay unsorted data\n");

write(st, n);

break;

case 3:

bubble1(st, n);

printf("\nDisplay of data sorted by descending ticket price \n");

write(st, n);

break;

case 4:

bubble2(st, n);

printf("\nDisplay of data sorted descending alphabetically by the name of the cinema\n");

write(st, n);

break;

case 5:

insertion1(st, n);

printf("\nDisplay of data sorted by ascendending ticket price\n");

write(st, n);

break;

case 6:

insertion2(st, n);

printf("\nDisplay of data sorted ascendending alphabetically by the name of the cinema\n");

write(st, n);

break;

case 0:

printf("Exit");

break;

default:

printf("Invalid choice.\n");

break;

}

} while (choice != 0);

return 0;

}

void read(struct Cinematograf \*st, int n) {

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

printf("\nThe name of the cinema : ");

scanf("%[^\n]%\*c", st[i].denumirea);

printf("Address : ");

scanf("%[^\n]%\*c", st[i].adresa);

printf("Phone : ");

scanf("%[^\n]%\*c", st[i].telefonul);

printf("Seat numbers :");

scanf("%[^\n]%\*c", st[i].nrlocuri);

printf("Ticket price : ");

scanf("%d", &st[i].pret);

clearBuffer();

}

}

void write(struct Cinematograf \*st, int n) {

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

printf("%s: %s, %s, %s, %d\n", st[i].denumirea, st[i].adresa, st[i].telefonul, st[i].nrlocuri, st[i].pret);

}

}

void clearBuffer() {

char c;

do {

c = getchar();

} while (c != '\n');

}

void bubble1(struct Cinematograf \*st, int n){

int i, schimbat;

struct Cinematograf \*p1, \*p2, aux;

p1 = st;

p2 = st + n - 1;

do {

schimbat = 0;

for(i = 0; i < n-1; i++, p1++){

if(p1->pret < (p1+1)->pret) {

aux = \*p1;

\*p1 = \*(p1+1);

\*(p1+1) = aux;

schimbat = 1;

}

}

n--;

p1 = st;

} while(schimbat);

}

void bubble2(struct Cinematograf \*st, int n){

int i, schimbat;

struct Cinematograf \*p1, \*p2, aux;

p1 = st;

p2 = st + n - 1;

do {

schimbat= 0;

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

if( strcmp( p1->denumirea, (p1+1)->denumirea) < 0){

aux = \*p1;

\*p1 = \*(p1+1);

\*(p1+1) = aux;

schimbat = 1; }

p1++; }

p1 = st;

n--;

} while(schimbat);

}

void insertion1(struct Cinematograf \*st, int n){

int i, j;

struct Cinematograf key;

for(i = 1; i < n; i++){

key = st[i];

j = i - 1;

while(j >= 0 && st[j].pret > key.pret){

st[j+1] = st[j];

j--;

}

st[j+1] = key;

}

}

void insertion2(struct Cinematograf \*st, int n){

int i, j;

struct Cinematograf key;

for(i = 1; i < n; i++){

key = st[i];

j = i - 1;

while(j >= 0 && strcmp(st[j].denumirea, key.denumirea) > 0){

st[j+1] = st[j];

j--;

}

st[j+1] = key;

}

}

void menu() {

printf("\nWelcome to menu:\n");

printf("1. Input data.\n");

printf("2. Display unsorted data\n");

printf("3. Display of data sorted by descending ticket price \n");

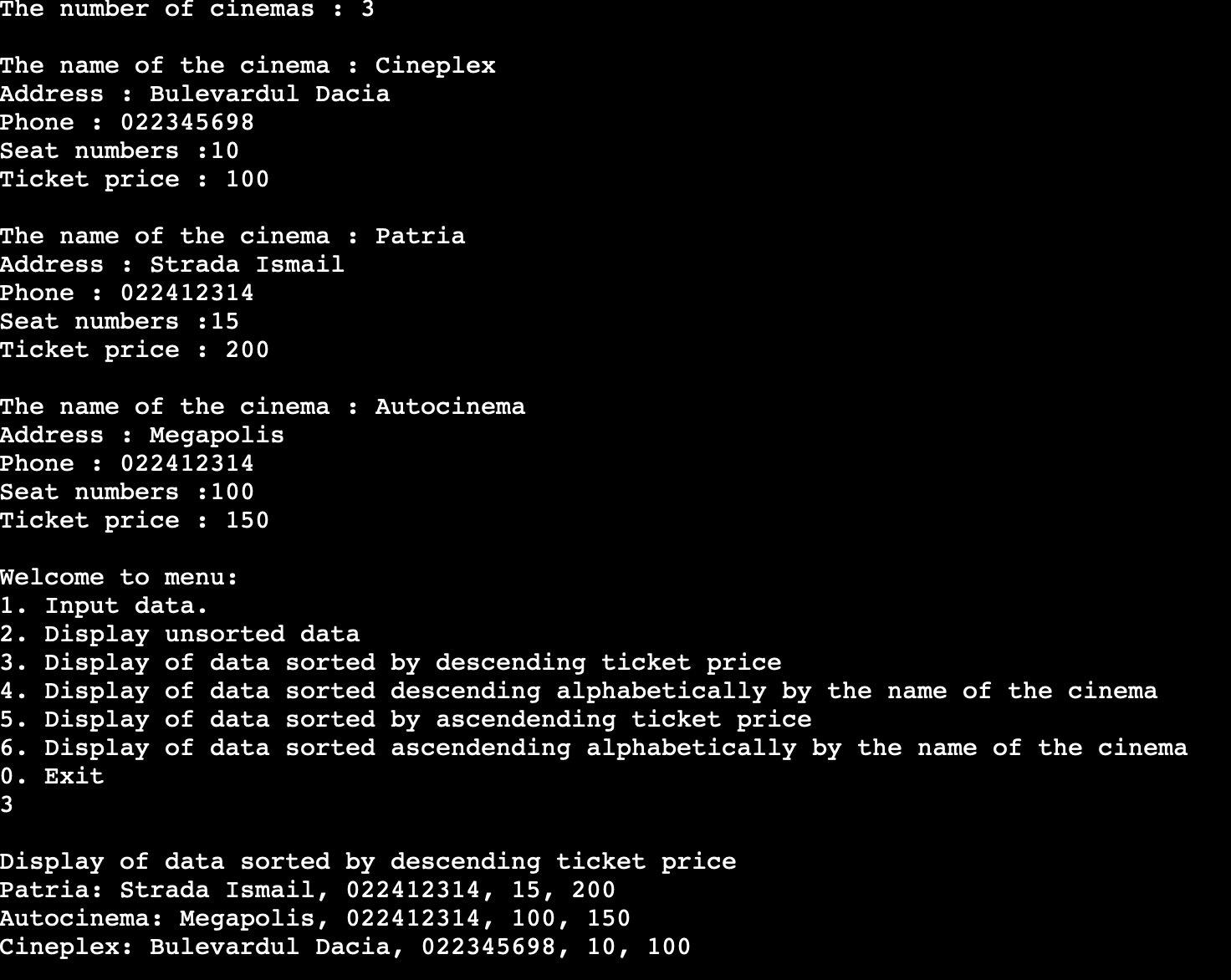
printf("4. Display of data sorted descending alphabetically by the name of the cinema\n");

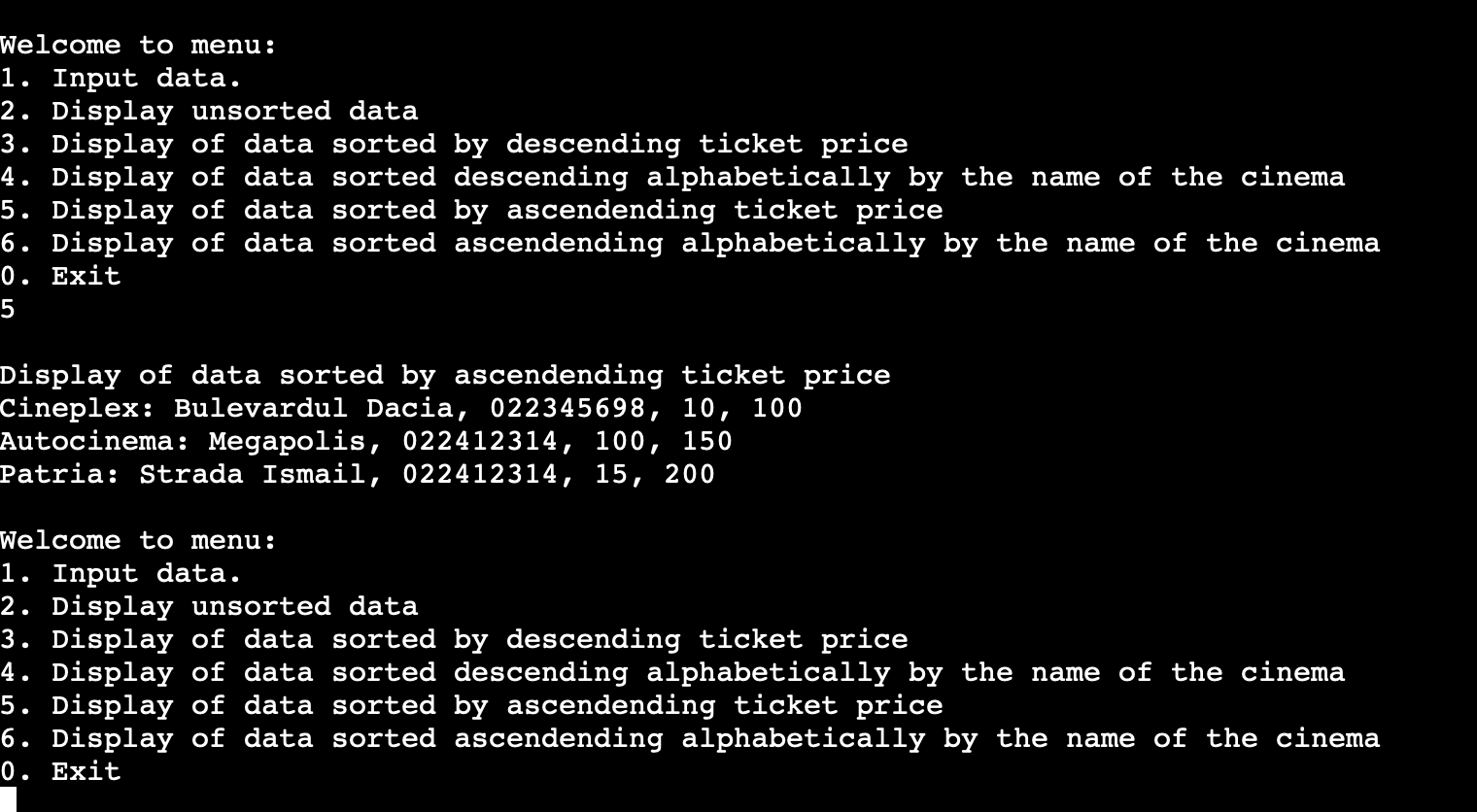
printf("5. Display of data sorted by ascendending ticket price\n");

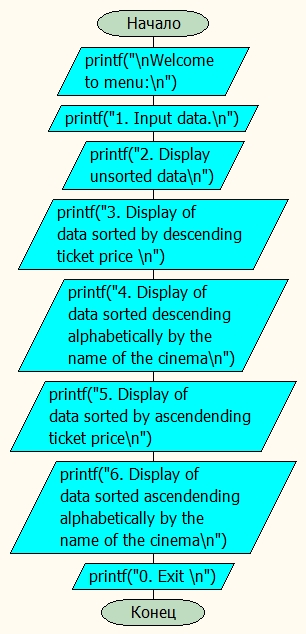
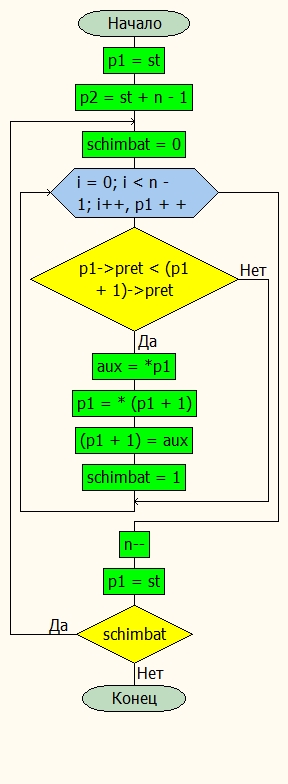
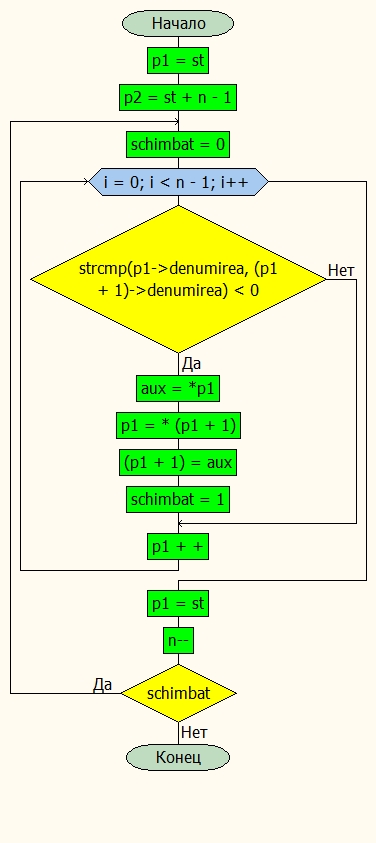
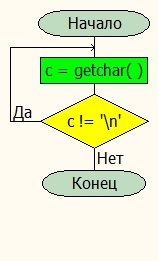
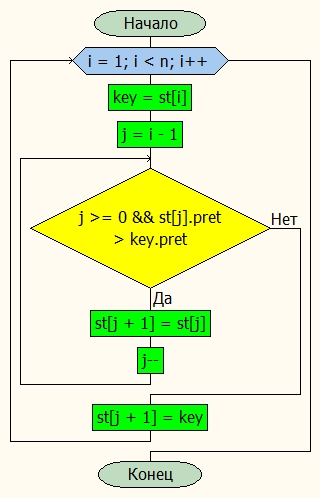
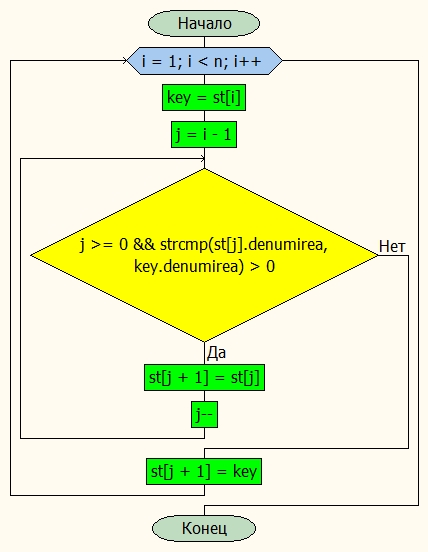
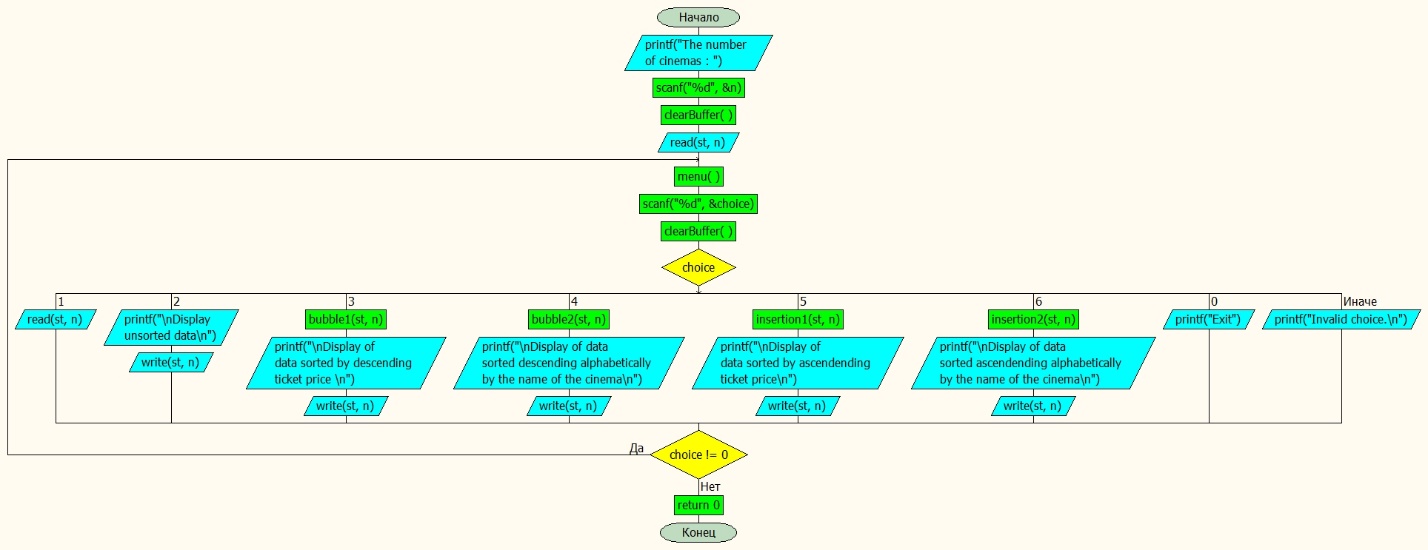
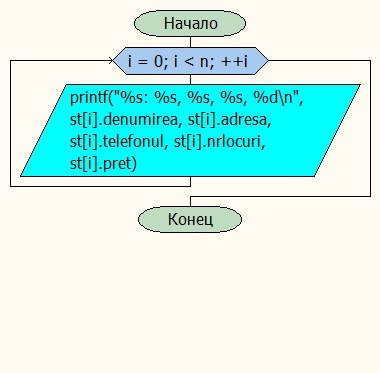
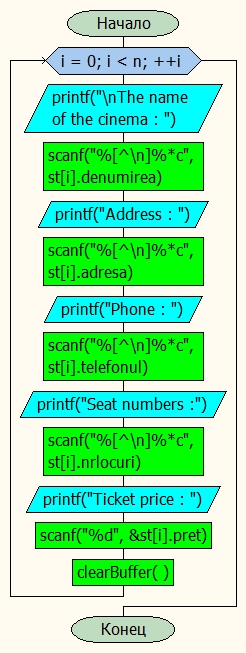
printf("6. Display of data sorted ascendending alphabetically by the name of the cinema\n");

printf("0. Exit \n");

}





****

**În concluzie,** programarea algoritmilor de prelucrare a structurilor și tablourilor de la structuri prin utilizarea funcțiilor, pointerilor și alocării dinamice a memoriei este un aspect esențial al programării în limbajul C. Aceste concepte sunt importante pentru a lucra eficient cu structuri de date complexe și pentru a crea algoritmi optimizați pentru diverse aplicații.

Utilizarea funcțiilor în programarea structurilor și a tablourilor de la structuri permite o abordare modulară, ușurând dezvoltarea, întreținerea și depanarea codului. Pointerii sunt necesari pentru a accesa și manipula structurile de date în mod eficient, iar alocarea dinamică a memoriei este necesară pentru a gestiona cu precizie cantitatea de memorie necesară pentru structurile de date în funcție de nevoile specifice ale aplicației.

În general, programarea în limbajul C necesită o atenție deosebită la detalii și o înțelegere solidă a conceptelor fundamentale. Dacă sunt utilizate corect, funcțiile, pointerii și alocarea dinamică a memoriei pot fi instrumente puternice pentru crearea de aplicații complexe și eficiente.