

# **Programarea Calculatoarelor și Limbaje de Programare II**

---

## **Profesor Coordonator**

Olteanu Gabriela

Neculoiu Georgian

## **Student**

Muraru Bianca-Maria

# **Sistem de rezervare bilete: un program pentru rezervarea și gestionarea biletelor.**

---

## **Profesor Coordonator**

Olteanu Gabriela

Neculoiu Georgian

## **Student**

Muraru Bianca-Maria

## Cuprins:

1.Introducere .....	4
2.Motivarea Alegerii Temei .....	4
2.1 Obiective propuse .....	5
2.2 Concepte și tehnologii utilizate în realizarea proiectului .....	6
3. Studiu de caz – Sistem de rezervare bilete.....	6
3.1 Tema aleasă .....	6
3.2 Implementarea pas cu pas .....	7
4. Concluzie .....	19
5. Bibliografie .....	20
6. Anexe .....	21

## Introducere

### Motivarea alegerii temei

Am ales să realizez o aplicație pentru vânzarea biletelor de tren, autocar și avion, implementată în limbajul C, deoarece consider că este o temă complexă, cu o aplicabilitate practică ridicată, care acoperă mai multe domenii importante ale dezvoltării software: gestiunea datelor, interacțiunea cu utilizatorul, logica aplicației și organizarea informației într-un mod eficient.

Trăim într-o lume în care mobilitatea este esențială, iar sistemele de rezervare au devenit parte integrantă a vieții de zi cu zi. De fiecare dată când călătorim, apelăm la astfel de aplicații pentru a ne planifica traseul, a alege cel mai potrivit mijloc de transport și a ne asigura locul. Din acest motiv, mi s-a părut o idee foarte bună să lucrez la un astfel de sistem, chiar dacă la o scară mai mică, pentru a înțelege mai bine ce presupune dezvoltarea unei aplicații utile și cu impact real asupra utilizatorilor.

Un alt motiv important pentru alegerea acestei teme este dorința mea de a-mi consolida cunoștințele de programare în C. Limbajul C este cunoscut ca fiind un limbaj de bază în informatică, cu o sintaxă strictă, care presupune o înțelegere clară a modului în care funcționează memoria, structurile de date și controlul fluxului programului. Realizarea acestui proiect a fost pentru mine o provocare, prin care am reușit să aplic noțiuni fundamentale într-un context real, să lucrez cu structuri, fișiere, meniuri și variabile, și să înțeleg mai bine ce înseamnă optimizarea codului și scrierea unui program.

Am fost atrasă și de componenta logică a temei. Gestionarea simultană a mai multor tipuri de transport (tren, autocar, avion), fiecare cu propriile particularități (ora, locuri, tipuri de bilete) m-a obligat să gândesc structurat. A fost important pentru mine ca utilizatorul final să poată accesa rapid și ușor toate funcționalitățile sistemului: consultarea curselor disponibile, rezervarea unui bilet, căutarea după diferite criterii etc. Astfel, am urmărit să dezvolt un program ușor de folosit, dar capabil să gestioneze date într-un mod coerent.

Nu în ultimul rând, am ales această temă pentru că reflectă interesul meu personal față de sistemele informatice care au un impact direct asupra oamenilor. Mi-a plăcut ideea de a crea un program care rezolvă o nevoie concretă: organizarea și rezervarea călătoriilor și care poate fi extins pe viitor cu noi funcționalități: plată online, conturi de utilizator, opțiuni de filtrare sau integrare cu baze de date. Această aplicație nu este doar un exercițiu tehnic, ci o primă încercare de a construi un produs util, de la zero, cu scopul de a fi folosit de alții.

În concluzie, am ales această temă pentru că îmbină partea practică, cu utilitate reală, cu provocările tehnice ale programării în C. Consider că, prin acest proiect, am avut ocazia să-mi dezvolt atât cunoștințele de programare, cât și capacitatea de a analiza cerințele unui sistem și de a le transpune într-o aplicație funcțională, coerentă și bine structurată.

### Obiective:

**Dezvoltarea unei aplicații în C pentru rezervare bilete -Mi-am propus să creez o aplicație care să permită rezervarea de bilete pentru tren, autocar și avion, fiecare cu un meniu și opțiuni clare pentru utilizator.**

- **Gestionarea eficientă a datelor** -Am urmărit să folosesc structuri de date potrivite pentru curse, locuri disponibile și rezervări, și să implementez salvarea acestora
- **Interacțiune ușoară cu utilizatorul** - Am dorit să construiesc un meniu simplu, text-based, prin care utilizatorul să poată căuta curse, rezerva locuri și să primească mesaje clare în caz de erori.
- **Organizarea logică și clară a codului** -Un obiectiv important a fost structurarea codului în funcții clare, ușor de înțeles, astfel încât aplicația să poată fi extinsă pe viitor.

### Tehnologii Utilizate:

În dezvoltarea acestei aplicații am utilizat limbajul de programare C, un limbaj esențial și de bază, care oferă un control precis asupra programului și resurselor hardware. Aplicația este implementată fără interfață grafică, astfel că totul se desfășoară în linia de comandă, printr-un meniu text, interacționându-se cu utilizatorul prin opțiuni simple de tipul „1. Cumpără un bilet”, „2. Vizualizează biletele cumpărate” și „3. Iesire”.

Am ales să construiesc aplicația într-un mod cât mai simplu, pentru a mă concentra pe logica funcțională, fără a adăuga complexitate inutilă. Toate datele utilizatorului, cum ar fi informațiile despre bilete, datele de călătorie, tipul de transport și destinațiile, sunt stocate folosind **tablouri** de tip `char` și `int`, ce reprezintă un mod eficient de a gestiona informațiile într-un program de dimensiuni mici.

Pentru implementarea funcționalității, am folosit librăriile standard din C:

- `stdio.h` pentru a efectua operațiile de intrare-ieșire (afișarea mesajelor în consolă, citirea datelor de la utilizator, și scrierea rezultatelor). Funcțiile `printf()` și `scanf()` sunt esențiale pentru interacțiunea cu utilizatorul.

- `string.h` pentru manipularea șirurilor de caractere, folosind funcții precum `strcpy()` pentru a copia textul în variabilele corespunzătoare și a-l afișa ulterior.

Un alt aspect important al aplicației este **gestionarea fișierelor**. Deși în acest cod nu am implementat salvarea datelor într-un fișier permanent, am utilizat variabilele de tip tablou pentru a păstra datele temporare, permițând utilizatorului să vizualizeze biletele cumpărate sau să efectueze noi rezervări pe parcurs .

Codul este structurat pentru a permite adăugarea de bilete și vizualizarea acestora într-un mod simplu. În plus, am folosit un sistem de verificare a erorilor, precum asigurarea că nu sunt depășite locurile disponibile și că datele introduse de utilizator sunt valide (ex: opțiuni de transport valide și număr de bilete disponibil).

### Studiu de caz – Tema aleasă

Tema aleasă pentru acest proiect, respectiv realizarea unei aplicații de rezervare bilete pentru transport (tren, autocar, avion), mi s-a părut nu doar interesantă, dar și foarte practică. În ziua de astăzi, procesul de achiziționare a билетelor pentru diverse mijloace de transport este un aspect esențial în viața noastră, iar dezvoltarea unei aplicații care să eficientizeze acest proces mi-a oferit ocazia de a învăța și aplica concepte importante de programare. Motivul pentru care am ales această temă este legat de dorința de a crea o aplicație care să răspundă unui tipar de utilizare real, aducând valoare utilizatorilor prin simplificarea unui proces frecvent întâlnit în viața de zi cu zi.

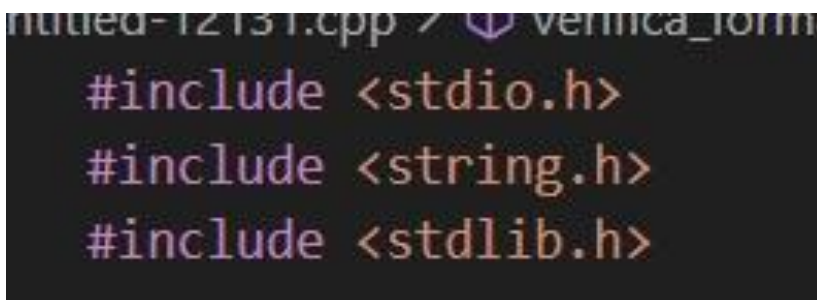
De asemenea, acest proiect mi-a permis să înțeleg mai bine cum funcționează gestionarea datelor într-o aplicație, cum să organizez și să gestionez informațiile într-un mod structurat și eficient. Un alt motiv pentru care am ales această temă este că, pentru a construi aplicația, am fost nevoită să aplic o serie de tehnici fundamentale ale programării, precum manipularea fișierelor, lucrul cu structuri de date (tablouri bidimensionale pentru stocarea datelor utilizatorului și a билетelor), și gestionarea fluxurilor de control (bucle și condiții). Aceste concepte mi-au permis să dezvolt o aplicație simplă, dar foarte funcțională, care poate fi ușor

adaptată și extinsă în viitor. Am considerat că această temă este și o provocare interesantă pentru mine, deoarece m-a ajutat să aplic cunoștințele dobândite până acum într-un context

practic, fără a apela la instrumente complexe sau la tehnologii care ar fi presupus un alt nivel de dificultate.

Alegerea unui sistem de rezervare bilete simplu mi-a oferit flexibilitate în proiectare și mi-a permis să înțeleg mai bine interacțiunea dintre utilizatori și sistemul informatic. De asemenea, am apreciat faptul că am avut posibilitatea să implementez o aplicație care simulează un sistem real, în care datele utilizatorilor sunt gestionate eficient și corect, iar aplicația oferă feedback instantaneu și informații precise. În acest fel, am putut să dezvolt un sistem ușor de utilizat, dar și unul care să respecte regulile și logica unui proces de rezervare real.

#### Implementarea pas cu pas:



```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
```

Biblioteca `stdio.h` (Standard Input/Output) este una dintre bibliotecile fundamentale ale limbajului C. Ea este folosită pentru gestionarea operațiunilor de input și output (I/O). În aplicația ta, această bibliotecă este folosită pentru a citi date de la utilizator și a afișa informații pe ecran.

Biblioteca `string.h` oferă funcții pentru manipularea șirurilor de caractere în C. În aplicația ta, această bibliotecă este utilizată pentru a lucra cu șiruri de caractere, cum ar fi numele utilizatorului, data călătoriei și destinațiile.

Biblioteca `stdlib.h` (standard library) oferă funcționalități de bază și utilitare în limbajul C, printre care se numără gestionarea memoriei, conversia datelor, generarea de numere aleatoare și funcții pentru terminarea programului. Este una dintre cele mai importante biblioteci standard din C.

- `#include <stdio.h>`: este folosită pentru **funcțiile de intrare și ieșire**
- `printf` – afișează text pe ecran

- scanf – citește date de la tastatură
- #include <string.h>: este folosită pentru **funcții care manipulează șiruri de caractere**, cum ar fi:
- strcpy() – copiază un șir într-altul
- strcmp() – compară două șiruri
- #include <stdlib.h>: este biblioteca standard care oferă:
- exit() – închide programul brusc în caz de eroare
- rand() – generează un număr aleator

```
int verifica_format_data(const char *data) {
    int zi, luna, an;
    if (scanf(data, "%d-%d-%d", &zi, &luna, &an) == 3 &&
        zi >= 1 && zi <= 31 &&
        luna >= 1 && luna <= 12 &&
        an >= 1000 && an <= 9999) {
        return 1;
    }
    return 0;
}
```

**int verifica\_format\_data(const char \*data) {**

int – tipul valorii returnate (1 = dată validă, 0 = dată invalidă).

verifica\_format\_data – numele funcției.

const char \*data – primește ca parametru un șir de caractere (ex: "12-05-2025"), adică data introdusă de utilizator.

**int zi, luna, an;**

1. Declară 3 variabile întregi în care vor fi stocate valorile extrase din șirul data: ziua, luna și anul.

**if (scanf(data, "%d-%d-%d", &zi, &luna, &an) == 3 &&**

scanf – funcție care extrage valori dintr-un șir de caractere.

"%d-%d-%d" – specifică faptul că șirul trebuie să conțină 3 numere întregi separate prin - (zi-lună-an).



&zi, &luna, &an – adresele variabilelor în care se salvează valorile extrase.

== 3 – verifică dacă au fost citite exact 3 valori (altfel formatul nu e corect).

**zi >= 1 && zi <= 31 &&**

Verifică dacă valoarea pentru zi este între 1 și 31 (o zi validă din lună)

**luna >= 1 && luna <= 12 &&**

Declarația și inițializarea variabilelor

```
int main() {
    char nume[MAX_BILETE][NUME_MAX];
    int numar_bilet[MAX_BILETE];
    char date[MAX_BILETE][20];
    char mijloc_transport[MAX_BILETE][20];
    char destinatii[MAX_BILETE][50];
    int numar_bilete_cumparate = 0;
    int optiune;
```

Aceasta este o directivă de preprocesare care definește constante simbolice.

MAX\_BILETE este setat la 10, ceea ce înseamnă că aplicația poate rezerva maximum 10 bilete. NUME\_MAX este setat la 50, ceea ce indică dimensiunea maximă a unui nume (50 de caractere).

### **Variabilele declarate :**

nume[MAX\_BILETE][NUME\_MAX]: Este un tablou bidimensional pentru a stoca numele persoanelor care cumpără bilete. Avem 10 bilete posibile, iar fiecare bilet poate avea un nume de maximum 50 de caractere.

numar\_bilet[MAX\_BILETE]: Stocază numerele biletelor pentru fiecare rezervare (de la 1 la 10).

date[MAX\_BILETE][20]: Stocază data fiecărei rezervări, cu formatul YYYY-MM-D

mijloc\_transport[MAX\_BILETE][20]: Stocază tipul de transport (Tren, Avion sau Autocar) pentru fiecare rezervare.

destinatii[MAX\_BILETE][50]: Stochează destinațiile alese de fiecare utilizator pentru fiecare bilet.

numar\_bilete\_cumparate: Un contor care ține evidența numărului total de bilete cumpărate.

optiune: Folosită pentru a stoca opțiunea aleasă de utilizator din meniul principal

```
while (1) {
    printf("\nSistem de cumpărare bilete\n");
    printf("1. Cumpără un bilet\n");
    printf("2. Vizualizează biletele cumpărate\n");
    printf("3. Iesire\n");
    printf("Alege o opțiune: ");
    scanf("%d", &optiune);

    if (optiune == 1) {
        if (numar_bilete_cumparate >= MAX_BILETE) {
            printf("Nu mai sunt locuri disponibile!\n");
            continue;
        }

        char data[20];
        printf("\nIntroduce data dorită pentru cumpărare : ");
        scanf("%s", data);

        if (!verifica_format_data(data)) {
            printf("Data NU este în formatul corect DD-MM-YYYY.\n");
            continue;
        }
    }
}
```

### while (1)

– bucla infinită ce menține programul activ și reafixează meniul până când utilizatorul alege opțiunea de ieșire ;

**printf("\nSistem de cumpărare bilete\n");**

– afișează titlul aplicației în consolă

**printf("1. Cumpără un bilet\n");**

– afișează opțiunea de cumpărare a unui bilet

**printf("2. Vizualizează biletele cumpărate\n");**

– afișează opțiunea pentru vizualizarea biletelor cumpărate

**printf("3. Iesire\n");**

– afișează opțiunea de ieșire din aplicație

**printf("Alege o opțiune: ");**

– solicită utilizatorului să introducă o opțiune

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

– citește opțiunea introdusă și o stochează în variabila „optiune”

**if (optiune == 1)**

– verifică dacă utilizatorul a ales opțiunea de cumpărare bilet

**if (numar\_bilete\_cumparate >= MAX\_BILETE)**

– verifică dacă s-a atins limita maximă de bilete care pot fi cumpărate

**printf("Nu mai sunt locuri disponibile!\n");**

– afișează un mesaj de eroare în cazul în care nu mai sunt bilete disponibile

**continue;**

– revine la începutul buclei fără a executa restul codului pentru această opțiune

**char data[20];**

– declară un șir de caractere pentru stocarea datei de călătorie

**printf("\nIntroduce data dorită pentru cumpărare : ");**

– cere utilizatorului să introducă data dorită pentru bilet

**scanf("%s", data);**

– citește data introdusă de utilizator

**if (!verifica\_format\_data(data))**

– verifică dacă data este în formatul corect (DD-MM-YYYY)

**printf("Data NU este în formatul corect DD-MM-YYYY.\n");**

– afișează un mesaj de eroare dacă data introdusă nu este validă

**continue;**

– revine la începutul buclei pentru a permite introducerea unei date corecte

```
int tip_transport;
printf("\nSelectează tipul de transport:\n");
printf("1. Tren\n");
printf("2. Avion\n");
printf("3. Autocar\n");
printf("Alege o opțiune: ");
scanf("%d", &tip_transport);

if (tip_transport < 1 || tip_transport > 3) {
    printf("Opțiune de transport invalidă.\n");
    continue;
}

int destinatie;
char transport[20];
char lista_destinatii[5][50];

if (tip_transport == 1) {
    strcpy(transport, "Tren");
    strcpy(lista_destinatii[0], "București - 08:00");
    strcpy(lista_destinatii[1], "Cluj - 09:30");
    strcpy(lista_destinatii[2], "Brașov - 11:00");
    strcpy(lista_destinatii[3], "Iași - 13:45");
    strcpy(lista_destinatii[4], "Timișoara - 17:20");
} else if (tip_transport == 2) {
    strcpy(transport, "Avion");
    strcpy(lista_destinatii[0], "Paris - 06:00");
    strcpy(lista_destinatii[1], "Londra - 08:15");
    strcpy(lista_destinatii[2], "Roma - 10:30");
    strcpy(lista_destinatii[3], "Berlin - 14:00");
    strcpy(lista_destinatii[4], "Madrid - 18:45");
} else {
    strcpy(transport, "Autocar");
}
```

**int tip\_transport;**

– declară o variabilă în care se va stoca opțiunea aleasă de utilizator pentru tipul de transport

**printf("\nSelectează tipul de transport:\n");**

– afișează mesajul introductiv pentru alegerea tipului de transport

**printf("1. Tren\n");**

– afișează opțiunea pentru transport cu trenul

**printf("2. Avion\n");**

– afișează opțiunea pentru transport cu avionul

**printf("3. Autocar\n");**

– afișează opțiunea pentru transport cu autocarul

**printf("Alege o opțiune: ");**

– solicită utilizatorului să introducă opțiunea dorită

**scanf("%d", &tip\_transport);**

– citește opțiunea aleasă de utilizator și o stochează în variabila tip\_transport

**if (tip\_transport < 1 || tip\_transport > 3)**

– verifică dacă opțiunea introdusă nu este validă (în afara intervalului 1–3)

**printf("Opțiune de transport invalidă.\n");**

– afișează un mesaj de eroare dacă opțiunea este incorectă

**continue;**

– revine la începutul buclei și permite o nouă introducere

**int destinatie;**

– declară variabila care va reține opțiunea aleasă pentru destinație

**char transport[20];**

– declară un șir de caractere care va reține tipul de transport selectat (ex: "Tren")

**char lista\_destinatii[5][50];**

– declară un tablou de 5 șiruri de caractere (fiecare de 50 caractere) pentru lista destinațiilor

**if (tip\_transport == 1)**

– verifică dacă utilizatorul a ales trenul ca mijloc de transport

**strcpy(transport, "Tren");**

– copiază valoarea "Tren" în variabila transport

**strcpy(lista\_destinatii[0], "București - 08:00");**

– setează prima destinație din lista pentru tren

**strcpy(lista\_destinatii[1], "Cluj - 09:30");**

– setează a doua destinație pentru tren

**strcpy(lista\_destinatii[2], "Brașov - 11:00");**

– setează a treia destinație pentru tren

**strcpy(lista\_destinatii[3], "Iași - 13:45");**

– setează a patra destinație pentru tren

**strcpy(lista\_destinatii[4], "Timișoara - 17:20");**

– setează a cincea destinație pentru tren

**else if (tip\_transport == 2)**

– verifică dacă utilizatorul a ales avionul

**strcpy(transport, "Avion");**

– copiază valoarea "Avion" în variabila transport

**strcpy(lista\_destinatii[0], "Paris - 06:00");**

– setează prima destinație din lista pentru avion

**strcpy(lista\_destinatii[1], "Londra - 08:15");**

– setează a doua destinație pentru avion

**strcpy(lista\_destinatii[2], "Roma - 10:30");**

– setează a treia destinație pentru avion

**strcpy(lista\_destinatii[3], "Berlin - 14:00");**

– setează a patra destinație pentru avion

**strcpy(lista\_destinatii[4], "Madrid - 18:45");**

– setează a cincea destinație pentru avion

**else**

– dacă opțiunea nu este 1 (tren) sau 2 (avion), atunci implicit este 3 (autocar)

**strcpy(transport, "Autocar");**

– copiază valoarea "Autocar" în variabila transport (urmează completarea cu destinațiile pentru autocar)

```
strcpy(transport, "Autocar");
strcpy(lista_destinatii[0], "Sibiu - 07:00");
strcpy(lista_destinatii[1], "Oradea - 10:00");
strcpy(lista_destinatii[2], "Constanța - 12:30");
strcpy(lista_destinatii[3], "Suceava - 15:15");
strcpy(lista_destinatii[4], "Craiova - 19:00");
}
printf("\nDestinații disponibile (%s):\n", transport);
for (int i = 0; i < 5; i++) {
    printf("%d. %s - Locuri disponibile: %d\n", i + 1, lista_destinatii[i], locuri_disponibile[tip_transport - 1][i]);
}
printf("Alege destinația: ");
scanf("%d", &destinatie);

if (destinatie < 1 || destinatie > 5) {
    printf("Destinație invalidă.\n");
    continue;
}
int nr_bilete;
printf("Câte bilete dorești să cumperi? ");
scanf("%d", &nr_bilete);

if (nr_bilete > locuri_disponibile[tip_transport - 1][destinatie - 1]) {
    printf("Nu sunt suficiente locuri disponibile pentru %d bilete.\n", nr_bilete);
    continue;
}
```

**strcpy(transport, "Autocar");**

– copiază valoarea „Autocar” în variabila transport, pentru a indica tipul de transport ales

**strcpy(lista\_destinatii[0], "Sibiu - 07:00");**

– setează prima destinație pentru autocar

```
strcpy(lista_destinatii[1], "Oradea - 10:00");
```

– setează a doua destinație pentru autocar

```
strcpy(lista_destinatii[2], "Constanța - 12:30");
```

– setează a treia destinație pentru autocar

```
strcpy(lista_destinatii[3], "Suceava - 15:15");
```

– setează a patra destinație pentru autocar

```
strcpy(lista_destinatii[4], "Craiova - 19:00");
```

– setează a cincea destinație pentru autocar

```
printf("\nDestinații disponibile (%s):\n", transport);
```

– afișează mesajul cu tipul de transport și lista destinațiilor aferente

```
for (int i = 0; i < 5; i++) {
```

– parcurge toate cele 5 destinații disponibile pentru transportul selectat

```
printf("%d. %s - Locuri disponibile: %d\n", i + 1, lista_destinatii[i],
```

```
locuri_disponibile[tip_transport - 1][i]);
```

– afișează numărul, denumirea destinației și numărul de locuri disponibile pentru fiecare

```
}
```

– încheie bucla for

```
printf("Alege destinația: ");
```

– cere utilizatorului să selecteze o destinație din lista afișată

```
scanf("%d", &destinatie);
```

– citește opțiunea introdusă și o salvează în variabila destinatie

```
if (destinatie < 1 || destinatie > 5) {
```

– verifică dacă destinația introdusă este în afara limitelor valide (1–5)

```
printf("Destinație invalidă.\n");
```

– afișează un mesaj de eroare dacă opțiunea nu este validă

```
continue;
```

– revine la începutul buclei principale și nu mai execută restul codului

```
}
```

– încheie blocul condiției

**int nr\_bilete;**

- declară variabila care va stoca numărul de bilete pe care utilizatorul dorește să le cumpere

**printf("Câte bilete dorești să cumperi? ");**

- afișează mesajul prin care solicită numărul de bilete dorite

**scanf("%d", &nr\_bilete);**

- citește valoarea introdusă de utilizator și o salvează în variabila nr\_bilete

**if (nr\_bilete > locuri\_disponibile[tip\_transport - 1][destinatie - 1]) {**

- verifică dacă numărul cerut de bilete depășește numărul de locuri disponibile pentru destinația selectată

**printf("Nu sunt suficiente locuri disponibile pentru %d bilete.\n", nr\_bilete);**

- afișează un mesaj de eroare dacă nu sunt suficiente locuri

**continue;**

- revine la începutul buclei, anulând tentativa de cumpărare curentă}
- încheie blocul condiției



```

    for (int i = 0; i < nr_bilete; i++) {
        int index = numar_bilete_cumparate + i;
        printf("Introduceți numele pentru biletul #%d: ", i + 1);
        scanf("%s", nume[index]);
        strcpy(date[index], data);
        strcpy(mijloc_transport[index], transport);
        strcpy(destinatii[index], lista_destinatii[destinatie - 1]);
        numar_bilet[index] = index + 1;
        printf("Bilet cumpărat cu succes! Număr bilet: %d\n", numar_bilet[index]);
    }
    numar_bilete_cumparate += nr_bilete;
    locuri_disponibile[tip_transport - 1][destinatie - 1] -= nr_bilete;
} else if (optiune == 2) {
    if (numar_bilete_cumparate == 0) {
        printf("Nu sunt bilete cumpărate.\n");
    } else {
        printf("\nBilete cumpărate:\n");
        for (int i = 0; i < numar_bilete_cumparate; i++) {
            printf("Biletul #%d [%s], la data de %s, cu %s către %s\n",
                numar_bilet[i], nume[i], date[i], mijloc_transport[i], destinatii[i]);
        }
    }
} else if (optiune == 3) {
    printf("La revedere!\n");
    break;
} else {
    printf("Opțiune invalidă!\n");
}

return 0;
}

```

for (int i = 0; i

**< nr\_bilete; i++) {**

– parcurge numărul de bilete pe care utilizatorul dorește să le cumpere

**int index = numar\_bilete\_cumparate + i;**

– calculează poziția exactă în vectori pentru fiecare bilet nou adăugat

**printf("Introduceți numele pentru biletul #%d: ", i + 1);**

– cere utilizatorului să introducă numele pentru fiecare bilet

**scanf("%s", nume[index]);**

– citește numele introdus și îl stochează în vectorul nume, la poziția curentă

**strcpy(date[index], data);**

– copiază data aleasă în vectorul care reține datele fiecărui bilet

**strcpy(mijloc\_transport[index], transport);**

– copiază tipul de transport ales în vectorul care stochează mijlocul de transport

**strcpy(destinatii[index], lista\_destinatii[destinatie - 1]);**

– copiază destinația selectată în vectorul destinatii

**numar\_bilet[index] = index + 1;**

– atribuie fiecărui bilet un număr unic, începând de la 1

**printf("Bilet cumpărat cu succes! Număr bilet: %d\n", numar\_bilet[index]);**

– afișează un mesaj de confirmare cu numărul biletului cumpărat

**}**

– încheie bucla for care prelucrează biletele

**numar\_bilete\_cumparate += nr\_bilete;**

– actualizează totalul biletelor cumpărate cu numărul tocmai procesat

**locuri\_disponibile[tip\_transport - 1][destinatie - 1] -= nr\_bilete;**

– scade din locurile disponibile numărul de bilete cumpărate pentru destinația și transportul ales

**else if (optiune == 2) {**

– verifică dacă utilizatorul a ales opțiunea 2: vizualizarea biletelor cumpărate

**if (numar\_bilete\_cumparate == 0) {**

– verifică dacă nu a fost cumpărat niciun bilet

**printf("Nu sunt bilete cumpărate.\n");**

– afișează un mesaj corespunzător dacă nu există bilete

**} else {**

– în caz contrar, dacă există bilete cumpărate

**printf("\nBilete cumpărate:\n");**

– afișează un titlu pentru lista de bilete

**for (int i = 0; i < numar\_bilete\_cumparate; i++) {**

– parcurge toate biletele cumpărate

**printf("Biletul #%-d – %-s, la data de %-s, cu %-s către %-s\n",**

numar\_bilet[i], nume[i], date[i], mijloc\_transport[i], destinatii[i]);

– afișează detaliile fiecărui bilet: număr, nume, dată, tip transport și destinație

**}**

– încheie bucla for

**}**

– încheie blocul else pentru opțiunea 2

```

else if (optiune == 3) {
    – verifică dacă utilizatorul a ales opțiunea 3: ieșire din aplicație

    printf("La revedere!\n");
    – afișează un mesaj de încheiere

    break;
    – întrerupe bucla while și iese din program

} else {
    – în cazul în care utilizatorul a introdus o opțiune invalidă

    printf("Opțiune invalidă!\n");
    – afișează un mesaj de eroare

}
    – încheie bucla principală while

return 0;
    – semnalează terminarea cu succes a programului

```

### Concluzie:

Acest proiect a reprezentat o experiență practică utilă pentru aprofundarea cunoștințelor în limbajul C, în special în ceea ce privește lucrul cu tablouri, condiții, funcții și structura unui program interactiv în consolă. Prin realizarea unui sistem de cumpărare bilete, am reușit să implementez funcționalități reale, cum ar fi validarea datelor introduse de utilizator, gestionarea locurilor disponibile și afișarea informațiilor într-un mod organizat. Acest proiect a reprezentat o experiență practică utilă pentru aprofundarea cunoștințelor în limbajul C, în special în ceea ce privește lucrul cu tablouri, condiții, funcții și structura unui program interactiv în consolă. Prin realizarea unui sistem de cumpărare bilete, am reușit să implementez funcționalități reale, cum ar fi validarea datelor introduse de utilizator, gestionarea locurilor disponibile și afișarea informațiilor într-un mod organizat.

### Bibliografie :

- [https://seriouscomputerist.atariverse.com/media/pdf/book/C%20Programming%20Language%20-%202nd%20Edition%20\(OCR\).pdf](https://seriouscomputerist.atariverse.com/media/pdf/book/C%20Programming%20Language%20-%202nd%20Edition%20(OCR).pdf)

- <https://archive.org/details/c-programming-a-modern-approach-2nd-ed-c-89-c-99-king-by>
- Wikipedia contributors. (n.d.). *C programming language*. Wikipedia. Retrieved from [\[https://en.wikipedia.org/wiki/C\\_\(programming\\_language\)\]](https://en.wikipedia.org/wiki/C_(programming_language))
- Tutorialspoint. (n.d.). *C Programming Language*. Retrieved from [\[https://www.tutorialspoint.com/cprogramming/index.htm\]](https://www.tutorialspoint.com/cprogramming/index.htm)
- Learn-C.org. (n.d.). *Learn C – Free Interactive C Tutorial*. Retrieved from [\[https://www.learn-c.org/\]](https://www.learn-c.org/)

## Anexe:

```
Sistem de cumpărare bilete
1. Cumpără un bilet
2. Vizualizează biletele cumpărate
3. Iesire
Alege o opțiune: 1

Introduce data dorită pentru cumpărare : 25-07-2026

Selectează tipul de transport:
1. Tren
2. Avion
3. Autocar
Alege o opțiune: 2

Destinații disponibile (Avion):
1. Paris - 06:00 - Locuri disponibile: 4
2. Londra - 08:15 - Locuri disponibile: 7
3. Roma - 10:30 - Locuri disponibile: 47
4. Berlin - 14:00 - Locuri disponibile: 33
5. Madrid - 18:45 - Locuri disponibile: 1
Alege destinația: 3
Câte bilete dorești să cumperi? 1
Introduceți numele pentru biletul #1: Bianca
Bilet cumpărat cu succes! Număr bilet: 1

Sistem de cumpărare bilete
1. Cumpără un bilet
2. Vizualizează biletele cumpărate
3. Iesire
Alege o opțiune: 2

Bilete cumpărate:
Biletul #1 - Bianca, la data de 25-07-2026, cu Avion către Roma - 10:30
```

Bilete cumpărate:

Biletul #1 – Bianca, la data de 25-07-2026, cu Avion către Roma - 10:30

Sistem de cumpărare bilete

1. Cumpără un bilet
2. Vizualizează biletele cumpărate
3. Iesire

Alege o opțiune: 3

La revedere!

PS D:\C++(pclp)\AIA\_LP-2025> █