

# Programarea calculatoarelor si limbaje de programare

Sistemul de management al informațiilor angajaților



Nume Student 1: **Torcea Ciprian-Marius**

Nume Student 2: **Văduva Alex**

Subgrupa: B

Grupa: CR 1.3

Profesor: Asist. Univ. Alexandra Vultureanu-Albisi

# Contents

<b>1</b>	<b>INTRODUCERE</b>	<b>3</b>
1.1	Enuntul problemei . . . . .	3
1.2	Descrierea problemei . . . . .	3
<b>2</b>	<b>ALGORITMI</b>	<b>4</b>
2.1	Pseudocod . . . . .	4
2.2	Scheme Logice . . . . .	6
<b>3</b>	<b>DESCRIEREA APLICATIEI</b>	<b>8</b>
3.1	Utilizare . . . . .	8
3.2	Avantaje . . . . .	9
3.3	Dezavantaje . . . . .	10
<b>4</b>	<b>REZULTATE</b>	<b>10</b>
<b>5</b>	<b>CONCLUZII</b>	<b>12</b>
<b>6</b>	<b>APPENDIX: PROGRAM C</b>	<b>12</b>
<b>7</b>	<b>BIBLIOGRAFIE</b>	<b>17</b>

# 1 INTRODUCERE

## 1.1 Enuntul problemei

Datele angajaților trebuie păstrate în orice companie. Fiecare companie are un angajat cu un ID unic de angajat, rol de angajat etc. Toate aceste date sunt păstrate într-un sistem de management al angajaților, unde sunt stocate toate datele despre fiecare angajat, putem prelua, actualiza și adăuga date la acest sistem. Folosind C putem crea un sistem de management al angajaților care poate îndeplini toate aceste sarcini, folosind cunoștințe de bază C precum șir, matrice, fișier etc.

**Funcționalitatea Sistemului de management al angajaților este:**

Realizați tabelul cu angajați.

Inserați intrări noi.

Ștergeți intrările.

Căutați o înregistrare.

## 1.2 Descrierea problemei

Problema abordată în acest program este gestionarea informațiilor despre angajații unei companii printr-un sistem de management al angajaților.

Scopul principal este să ofere o interfață utilizatorului pentru a adăuga, șterge, căuta și afișa informații despre angajați. Acest sistem utilizează un set de funcționalități de bază precum afișarea, adăugarea, ștergerea și căutarea angajaților într-un tabel.

*Principalele componente ale problemei sunt:*

**Structura de date:** Un angajat este reprezentat printr-o structură care conține un ID unic, nume și rol în cadrul companiei.

**Funcționalități principale:**

1. Afișarea tabelului cu angajați.
2. Adăugarea unei noi înregistrări pentru un angajat.
3. Ștergerea unei înregistrări bazată pe ID-ul angajatului.
4. Căutarea unui angajat în funcție de ID.

**Persistența datelor:**

Datele despre angajați pot fi salvate și încărcate dintr-un fișier pentru a asigura persistența informațiilor între rulările programului.

## 2 ALGORITMI

### 2.1 Pseudocod

#### Pseudocod pentru Funcția printMenu

- 
- 1: Afișează textul pentru meniu
  - 2: Încheie funcția
- 

#### Pseudocod pentru Funcția printEmployee

- 
- 1: Afișează informațiile despre angajat (ID, nume, rol)
  - 2: Încheie funcția
- 

#### Pseudocod pentru Funcția addEmployee

- 
- 1: **if** count < MAX\_EMPLOYEES **then**
  - 2:     Afișează un mesaj de introducere
  - 3:     Citește ID-ul, numele și rolul pentru noul angajat
  - 4:     Incrementează count
  - 5:     Afișează un mesaj de succes
  - 6: **else**
  - 7:     Afișează un mesaj că numărul maxim de angajați a fost atins
  - 8: **end if**
  - 9: Încheie funcția
- 

#### Pseudocod pentru Funcția displayEmployees

---

```
1: Afisează un antet pentru tabelul de angajați
2: for all angajat în angajați do
3:     Apelează PRINTEMPLOYEE cu angajatul curent
4: end for
5: Afisează un separator pentru tabel
6: Încheie funcția
```

---

## Pseudocod pentru Funcția deleteEmployee

---

```
1: Inițializează index cu -1
2: for all angajat în angajați do
3:     if angajatul curent are ID-ul dorit then
4:         Setează index cu indicele curent
5:         Ieșește din bucla for
6:     end if
7: end for
8: if index  $\neq$  -1 then
9:     for i de la index la count - 1 do
10:        Copiază angajatul de la i+1 la i
11:    end for
12:    Decrementează count
13:    Afisează un mesaj de succes
14: else
15:    Afisează un mesaj că angajatul nu a fost găsit
16: end if
17: Încheie funcția
```

---

## Pseudocod pentru Funcția searchEmployee

---

```
1: Inițializează found cu 0
2: for all angajat în angajați do
3:     if angajatul curent are ID-ul dorit then
4:         Apelează PRINTEMPLOYEE cu angajatul curent
5:         Setează found la 1
6:         Ieșește din bucla for
7:     end if
8: end for
9: if found == 0 then
10:    Afisează un mesaj că angajatul nu a fost găsit
11: end if
12: Încheie funcția
```

---

## Pseudocod pentru Funcția saveToFile

---

```
1: Deschide un fișier pentru scriere (file)
2: if file nu există then
3:     Afișează un mesaj de eroare
4:     Încheie funcția
5: end if
6: for all angajat în angajați do
7:     Scrie informațiile despre angajat în file
8: end for
9: Închide file
10: Afișează un mesaj de succes
11: Încheie funcția
```

---

## Pseudocod pentru Funcția loadFromFile

---

```
1: Deschide un fișier pentru citire (file)
2: if file nu există then
3:     Afișează un mesaj că fișierul nu poate fi deschis
4:     Încheie funcția
5: end if
6: while se poate citi din file do
7:     Citește ID-ul, numele și rolul pentru un angajat nou
8:     Incrementează count
9:     if count  $\geq$  MAX_EMPLOYEES then
10:         Afișează un mesaj că numărul maxim de angajați a fost atins
11:         Încheie bucla while
12:     end if
13: end while
14: Închide file
15: Afișează un mesaj de succes
16: Încheie funcția
```

---

## 2.2 Scheme Logice

Meniul aplicației noastre de management al angajaților oferă mai multe opțiuni, iar pentru fiecare opțiune am implementat o metodă specifică, aplicația fiind modulară.

În următoare figură sunt evidențiate opțiunile disponibile ale meniului:

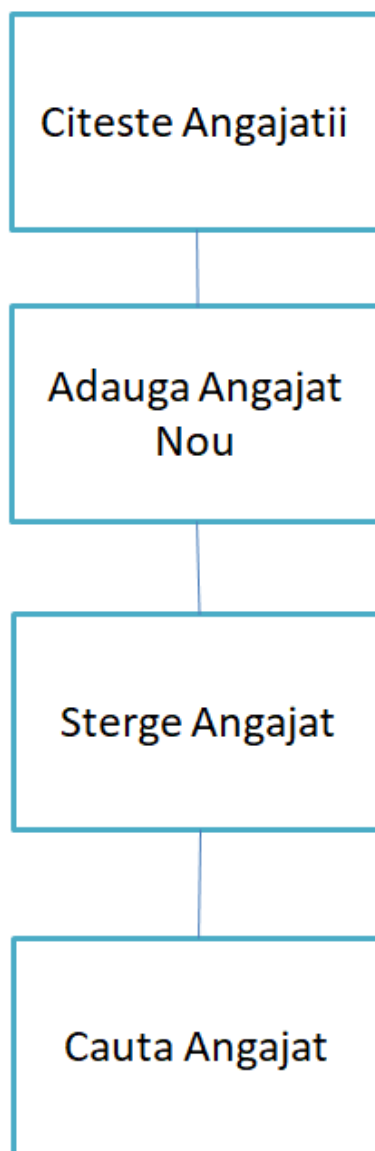


Figure 1: Optiuni Meniu

## 3 DESCRIEREA APLICATIEI

### 3.1 Utilizare

Aplicația de management al angajaților este o unealtă simplă dezvoltată în limbajul C, care permite utilizatorilor să gestioneze informațiile despre angajați într-un mod interactiv.

***O scurtă descriere a utilizării aplicației:***

#### **Lansarea Aplicației:**

Utilizatorul lansează aplicația de la consolă sau terminal.

#### **Afișarea Meniului:**

La început, aplicația afișează un meniu interactiv care oferă diferite opțiuni. Utilizatorul poate vedea tabelul cu angajați, adăuga un nou angajat, șterge un angajat, caută un angajat sau să iasă din aplicație.

#### **Afișarea Angajaților:**

Opțiunea "Afișati tabelul cu angajați" permite utilizatorului să vadă informațiile despre toți angajații existenți în sistem.

#### **Adăugarea unui Angajat Nou:**

Opțiunea "Adăugați o nouă înregistrare" permite utilizatorului să introducă datele unui angajat nou, inclusiv ID-ul, numele și rolul.

#### **Ștergerea unui Angajat:**

Opțiunea "Ștergeți o înregistrare" permite utilizatorului să șteargă un angajat existent în funcție de ID-ul acestuia.

#### **Căutarea unui Angajat:**

Opțiunea "Căutați o înregistrare" permite utilizatorului să introducă ID-ul unui angajat și să afle informațiile despre acel angajat.

#### **Ieșire din Aplicație:**

Opțiunea "Ieșiți" încheie aplicația și închide meniul.

#### **Salvarea și Încărcarea Datelor:**

Datele angajaților sunt salvate într-un fișier la închiderea aplicației și încărcate din acel fișier la pornirea aplicației.

#### **Validări:**

Aplicația oferă feedback și mesaje de eroare pentru a asigura că utilizatorul introduce date valide și că operațiile sunt efectuate corect.

#### **Interacțiune Continuă:**

Utilizatorul poate interacționa cu aplicația într-un mod iterativ, efectuând diferite operații până când decide să iasă din aplicație.

Această aplicație simplă oferă un mediu interactiv pentru gestionarea datelor despre angajați și permite utilizatorilor să facă diverse operații asupra acestora.



## 3.2 Avantaje

Utilizarea aplicației de management al angajaților dezvoltată în limbajul C oferă mai multe avantaje, în special pentru gestionarea și monitorizarea eficientă a informațiilor despre angajați într-o companie sau organizație.

### *Câteva dintre avantajele utilizării acestei aplicații:*

#### **Ușurința Utilizării:**

Interfața simplă și meniul interactiv fac aplicația ușor de utilizat, chiar și pentru persoanele fără experiență în programare.

**Gestionarea Eficientă a Angajaților:** Utilizatorii pot adăuga, șterge, căuta și vizualiza informații despre angajați, facilitând gestionarea resurselor umane în organizație.

#### **Salvare și Încărcare Automată a Datelor:**

Datele despre angajați sunt salvate automat la închiderea aplicației și încărcate la deschiderea acesteia, oferind consistență și persistență datelor.

**Rezolvarea Rapidă a Problemei:** Utilizatorii pot identifica rapid angajați, adăuga noi informații sau șterge datele existente, rezolvând rapid problemele legate de informațiile despre angajați.

#### **Feedback și Validare:**

Aplicația oferă feedback utilizatorului și validează datele introduse, asigurându-se că acestea sunt corecte și complete.

#### **Automatizarea Proceselor Repetitive:**

Aplicația permite automatizarea unor procese repetitive legate de gestionarea angajaților, cum ar fi adăugarea sau ștergerea acestora.

#### **Portabilitate și Accesibilitate:**

Dezvoltată în limbajul C, aplicația poate fi compilată și rulată pe diverse platforme, oferind portabilitate și accesibilitate.

#### **Personalizare Ușoară:**

Utilizatorii pot modifica ușor codul sursă pentru a adăuga funcționalități suplimentare sau pentru a personaliza aplicația în funcție de nevoile specifice ale organizației.

#### **Învățare și Dezvoltare:**

Aplicația oferă o oportunitate pentru cei care doresc să învețe sau să dezvolte abilități în programare în limbajul C.

#### **Flexibilitate în Utilizare:**

Utilizatorii pot executa diferite operații într-o ordine flexibilă, adaptând aplicația la nevoile lor specifice.

### 3.3 Dezavantaje

Deși aplicația de management al angajaților dezvoltată în limbajul C are numeroase avantaje, există și câteva dezavantaje, cum ar fi:

#### **Interfață Textuală:**

Interfața text-based poate să nu fie la fel de atrăgătoare sau intuitivă pentru toți utilizatorii, în special pentru cei obișnuiți cu aplicații grafice.

#### **Limitări în Extensibilitate:**

Extinderea funcționalității sau adăugarea de caracteristici noi poate fi mai dificilă în comparație cu aplicațiile dezvoltate în limbaje mai moderne și orientate pe obiect.

#### **Gestionare Manuală a Fișierelor:**

Aplicația salvează și încarcă datele în mod manual dintr-un fișier text, ceea ce poate fi nepractic într-un mediu unde gestionarea datelor se face în baze de date mai avansate.

#### **Lipsa Securității Avansate:**

Aplicația nu dispune de caracteristici avansate de securitate, cum ar fi autentificarea utilizatorilor sau criptarea datelor, ceea ce poate fi o vulnerabilitate în medii sensibile.

#### **Limitări în Manipularea Datelor:**

Manipularea datelor este limitată la operațiuni de bază (adică adăugare, ștergere, căutare), iar operațiuni complexe sau filtrare avansată pot să lipsească.

#### **Dependență de Cunoștințe Tehnice:**

Utilizarea eficientă a aplicației poate necesita cunoștințe minime de programare în limbajul C, ceea ce poate exclude anumite categorii de utilizatori.

#### **Lipsa Interactivității Avansate:**

Nu oferă funcționalități interactive avansate sau interfețe grafice, ceea ce poate limita experiența utilizatorului în comparație cu aplicațiile moderne.

#### **Gestionarea Manuală a Memoriei:**

Dezvoltarea în limbajul C implică gestionarea manuală a memoriei, ceea ce poate duce la posibile erori de memorie sau scurgeri dacă nu este gestionată corespunzător.

## 4 REZULTATE

Utilizatorii pot efectua o varietate de operațiuni asupra acestor date, inclusiv vizualizarea și actualizarea tabelului cu angajați, adăugarea unor noi intrări, ștergerea sau căutarea de înregistrări specifice. Astfel, oferă un mediu practic și accesibil pentru administrarea eficientă a detaliilor legate de personal în cadrul unei organizații.


Figura următoare arată meniul cu opțiunile disponibile și fișierul ce conține datele introduse de către utilizator:

```
Datele au fost incarcate cu succes din fisier!
```

```
===== Meniu =====
1. Afisati tabelul cu angajati
2. Adaugati o noua inregistrare
3. Stergeti o inregistrare
4. Cautati o inregistrare
5. Iesiti
=====
Alegeti o optiune: 1

===== Tabelul cu angajati =====
ID: 1 | Nume: torcea | Rol: student
ID: 2 | Nume: vaduva | Rol: student
=====

===== Meniu =====
1. Afisati tabelul cu angajati
2. Adaugati o noua inregistrare
3. Stergeti o inregistrare
4. Cautati o inregistrare
5. Iesiti
=====
Alegeti o optiune:
```

 employee\_data - Notepad

File Edit Format View Help

1 torcea student

2 vaduva student

Figure 2: Inserare și afișare date angajați

## 5 CONCLUZII

În urma dezvoltării acestei aplicații simple de management al angajaților în limbajul C, putem trage câteva concluzii relevante:

### **Simplu și Eficient:**

Aplicația oferă o soluție simplă și eficientă pentru gestionarea informațiilor despre angajați, utilizând un limbaj de programare precum C.

### **Structură Modulară:**

Implementarea modulară, cu funcții dedicate pentru fiecare operațiune, facilitează înțelegerea și întreținerea codului.

### **Interactivitate:**

Meniul interactiv permite utilizatorilor să interacționeze ușor cu aplicația, efectuând diverse operațiuni asupra datelor despre angajați.

### **Persistența Datelor:**

Salvarea și încărcarea automată a datelor într-un fișier text asigură persistența informațiilor între diferite rulări ale aplicației.

### **Limitări și Dezavantaje:**

Totuși, aplicația are și limitările sale, cum ar fi lipsa caracteristicilor avansate și a interfeței grafice.

## 6 APPENDIX: PROGRAM C

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <string.h>
4
5  #define MAX_EMPLOYEES 100
6  #define MAX_NAME_LENGTH 50
7
8  // Structura pentru un angajat
9  struct Employee
10 {
11     int id;
12     char name[MAX_NAME_LENGTH];
13     char role[MAX_NAME_LENGTH];
14 };
15
16 // Functie pentru afisarea meniului
17 void printMenu()
18 {
19     printf("\n==== Meniu =====\n");
20     printf("1. Afisati tabelul cu angajati\n");
21     printf("2. Adaugati o noua inregistrare\n");
22     printf("3. Stergeti o inregistrare\n");
23     printf("4. Cautati o inregistrare\n");
```

```

24     printf("5. Iesiti\n");
25     printf("=====\n");
26 }
27
28 // Functie pentru afisarea unui angajat
29 void printEmployee(struct Employee emp)
30 {
31     printf("ID: %d | Nume: %s | Rol: %s\n", emp.id, emp.name, emp.
        role);
32 }
33
34 // Functie pentru afisarea intregului tabel de angajati
35 void displayEmployees(struct Employee employees[], int count)
36 {
37     printf("\n==== Tabelul cu angajati ==== \n");
38     for (int i = 0; i < count; ++i)
39     {
40         printEmployee(employees[i]);
41     }
42     printf("==== \n");
43 }
44
45 // Functie pentru adaugarea unui nou angajat
46 void addEmployee(struct Employee employees[], int *count)
47 {
48     if (*count < MAX_EMPLOYEES)
49     {
50         printf("\nIntroduceti datele noului angajat:\n");
51
52         printf("ID: ");
53         scanf("%d", &employees[*count].id);
54
55         printf("Nume: ");
56         scanf("%s", employees[*count].name);
57
58         printf("Rol: ");
59         scanf("%s", employees[*count].role);
60
61         (*count)++;
62         printf("Angajat adaugat cu succes!\n");
63     }
64     else
65     {
66         printf("Numarul maxim de angajati a fost atins!\n");
67     }
68 }
69
70 // Functie pentru stergerea unui angajat dupa ID
71 void deleteEmployee(struct Employee employees[], int *count, int id)
72 {
73     int index = -1;

```

```

74     for (int i = 0; i < *count; ++i)
75     {
76         if (employees[i].id == id)
77         {
78             index = i;
79             break;
80         }
81     }
82
83     if (index != -1)
84     {
85         for (int i = index; i < *count - 1; ++i)
86         {
87             employees[i] = employees[i + 1];
88         }
89         (*count)--;
90         printf("Angajat sters cu succes!\n");
91     }
92     else
93     {
94         printf("Angajatul cu ID-ul %d nu a fost gasit!\n", id);
95     }
96 }
97
98 // Functie pentru cautarea unui angajat dupa ID
99 void searchEmployee(struct Employee employees[], int count, int id)
100 {
101     int found = 0;
102     for (int i = 0; i < count; ++i)
103     {
104         if (employees[i].id == id)
105         {
106             printEmployee(employees[i]);
107             found = 1;
108             break;
109         }
110     }
111
112     if (!found)
113     {
114         printf("Angajatul cu ID-ul %d nu a fost gasit!\n", id);
115     }
116 }
117
118 // Functie pentru salvarea datelor intr-un fisier
119 void saveToFile(struct Employee employees[], int count)
120 {
121     FILE *file = fopen("employee_data.txt", "w");
122     if (file == NULL)
123     {
124         printf("Eroare la deschiderea fisierului!\n");

```

```

125         return;
126     }
127
128     for (int i = 0; i < count; ++i)
129     {
130         fprintf(file, "%d %s %s\n", employees[i].id, employees[i].
            name, employees[i].role);
131     }
132
133     fclose(file);
134     printf("Datele au fost salvate cu succes in fisier!\n");
135 }
136
137 // Functie pentru incarcarea datelor dintr-un fisier
138 void loadFromFile(struct Employee employees[], int *count)
139 {
140     FILE *file = fopen("employee_data.txt", "r");
141     if (file == NULL)
142     {
143         printf("Fisierul nu exista sau nu poate fi deschis!\n");
144         return;
145     }
146
147     while (fscanf(file, "%d %s %s", &employees[*count].id, employees
        [*count].name, employees[*count].role) != EOF)
148     {
149         (*count)++;
150         if (*count >= MAX_EMPLOYEES)
151         {
152             printf("Ati atins numarul maxim de angajati. Incarcarea s
                -a oprit.\n");
153             break;
154         }
155     }
156
157     fclose(file);
158     printf("Datele au fost incarcate cu succes din fisier!\n");
159 }
160
161 int main()
162 {
163     struct Employee employees[MAX_EMPLOYEES];
164     int employeeCount = 0;
165     int choice;
166
167     loadFromFile(employees, &employeeCount);
168
169     do
170     {
171         printMenu();
172         printf("Alegeti o optiune: ");

```

```

173     scanf("%d", &choice);
174
175     switch (choice)
176     {
177     case 1:
178         displayEmployees(employees, employeeCount);
179         break;
180     case 2:
181         addEmployee(employees, &employeeCount);
182         break;
183     case 3:
184         if (employeeCount > 0)
185         {
186             int deleteId;
187             printf("Introduceti ID-ul angajatului de sters: ");
188             scanf("%d", &deleteId);
189             deleteEmployee(employees, &employeeCount, deleteId);
190         }
191         else
192         {
193             printf("Nu exista angajati de sters!\n");
194         }
195         break;
196     case 4:
197         if (employeeCount > 0)
198         {
199             int searchId;
200             printf("Introduceti ID-ul angajatului de cautat: ");
201             scanf("%d", &searchId);
202             searchEmployee(employees, employeeCount, searchId);
203         }
204         else
205         {
206             printf("Nu exist angajati de cautat!\n");
207         }
208         break;
209     case 5:
210         saveToFile(employees, employeeCount);
211         printf("La revedere!\n");
212         break;
213     default:
214         printf("Optiune invalida! Alegeti din nou.\n");
215     }
216 }
217 while (choice != 5);
218
219 return 0;
220 }

```



## 7 BIBLIOGRAFIE

### References

- [1] Overleaf Documentation, <https://www.overleaf.com/learn>.
- [2] Overleaf Tutorials, <https://www.overleaf.com/learn/latex/Tutorials>.
- [3] Employee Management System Project using C, <https://www.studytonight.com/c-projects/employee-management-system-project-using-c-language>.
- [4] Employee Record System in C using File Handling, <https://www.geeksforgeeks.org/employee-record-system-in-c-using-file-handling/>.
- [5] C Tutorials, <https://www.w3schools.com/c/>.
- [6] Menu-Driven program using Switch-case in C, <https://www.geeksforgeeks.org/menu-driven-program-using-switch-case-c/>.