



Royaume du Maroc
Ministère de l'Enseignement Supérieur, de la Recherche Scientifique et de
l'Innovation



Université Mohammed V
Ecole Normale Supérieure de Rabat (ENS)
2023-2024



Gestion des employés en Langage C

Réalisé par :

MOUEDDENE Amina

MOUEDDENE Wafae

HACHIMI ALAOUI Chaimae

Demandé par :

Pr EL MOUNADI Abdelali

Année universitaire 2023/2024

SOMMAIRE

Introduction	4
Définition des Structures de Données	5
Chargement et Mise à Jour du Fichier.....	5
Menu d'Opérations	6
a) Ajouter de Nouveaux Employés.....	6
b) Supprimer des Employés Existants.....	7
c) Rechercher un Employé	7
d) Mettre à Jour les Informations d'un Employé	8
e) Enregistrer les Informations dans le Fichier.....	9
f) Quitter le Programme.....	9
Partie Console :	10
Menu de choix :	10
L'opération d'affichage des employés :	10
L'opération d'ajout d'un employé :	11
L'opération de suppression par code :	11
Conclusion	12

La liste des figures

Figure 1:Les structures utilisés	5
Figure 2:Le code de chargement du fichier.....	6
Figure 3:Le code d'ajout d'un nouvel employé	6
Figure 4:Le code de suppression de l'employé	7
Figure 5: Le code de rechercher l'employé par code	7
Figure 6: Le code de rechercher l'employé par nom.....	8
Figure 7: Le code de rechercher l'employé par prénom	8
Figure 8: Le code de modifier l'employé par code	9
Figure 9: Le code de sauvegarde	9
Figure 10: Le menu d'affichage	10
Figure 11: La liste d'employés	10
Figure 12: L'ajout d'un nouvel employé	11
Figure 13: La suppression par code	11

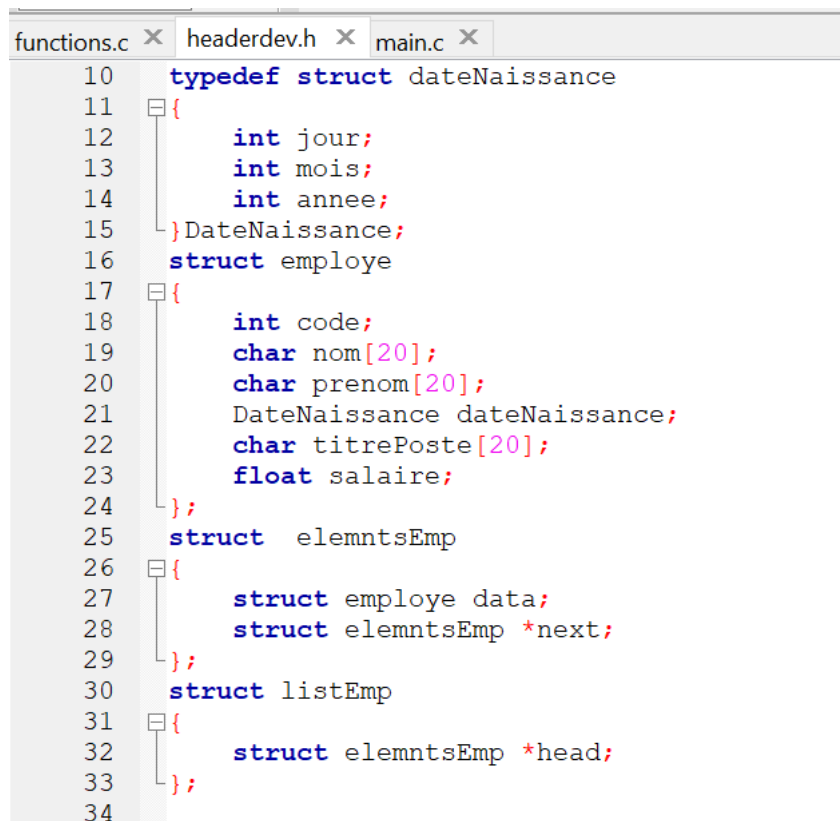
Introduction

Le projet consiste en la conception d'un programme en langage C pour la gestion d'une liste d'employés. Chaque employé est caractérisé par un code unique, son nom de famille, son prénom, sa date de naissance, son titre de poste, et son salaire. L'objectif est de créer un système robuste permettant l'ajout, la suppression, la recherche, la mise à jour et la sauvegarde d'informations sur les employés.

Définition des Structures de Données

Dans le cadre de ce projet, des structures de données ont été définies pour représenter un employé. Le langage C offre la flexibilité nécessaire pour décrire ces structures de manière précise :

La structure « **dateNaissance** » comprend des champs pour stocker le jour, le mois et l'année pour chaque employé. Ainsi, la structure « **employe** » qui comprend des champs pour stocker toutes les informations pertinentes pour chaque employé.



```
functions.c x headerdev.h x main.c x
10 typedef struct dateNaissance
11 {
12     int jour;
13     int mois;
14     int annee;
15 } DateNaissance;
16 struct employe
17 {
18     int code;
19     char nom[20];
20     char prenom[20];
21     DateNaissance dateNaissance;
22     char titrePoste[20];
23     float salaire;
24 };
25 struct elemntsEmp
26 {
27     struct employe data;
28     struct elemntsEmp *next;
29 };
30 struct listEmp
31 {
32     struct elemntsEmp *head;
33 };
34
```

Figure 1: Les structures utilisés

Chargement et Mise à Jour du Fichier

Le programme débute par le chargement du contenu du fichier "employees.txt" au moment de son lancement. Cette approche est choisie pour minimiser les opérations de lecture/écriture coûteuses en mémoire. À la fin de l'exécution, le programme met à jour le fichier avec les éventuelles modifications apportées aux données des employés.

```

functions.c x headerdev.h x main.c x
28
29 struct listEmp* initiList()
30 {
31     struct listEmp *list = malloc(sizeof(struct listEmp));
32     list->head = NULL;
33
34     return list;
35
36 };
37
38 int estVide(struct listEmp* list)
39 {
40     return list->head == NULL;
41 }
42
43 void afficherList(struct listEmp* list)
44 {
45     if(estVide(list))
46     {
47         printf("la liste est vide. \n");
48         return;
49     }
50
51     struct elemntsEmp* it = list->head;
52     while(it != NULL)
53     {

```

Figure 2:Le code de chargement du fichier

Menu d'Opérations

Le programme propose un menu interactif offrant diverses fonctionnalités :

a) Ajouter de Nouveaux Employés

L'utilisateur peut ajouter de nouveaux employés en fournissant les informations nécessaires. Le programme génère un code unique pour chaque nouvel employé.

```

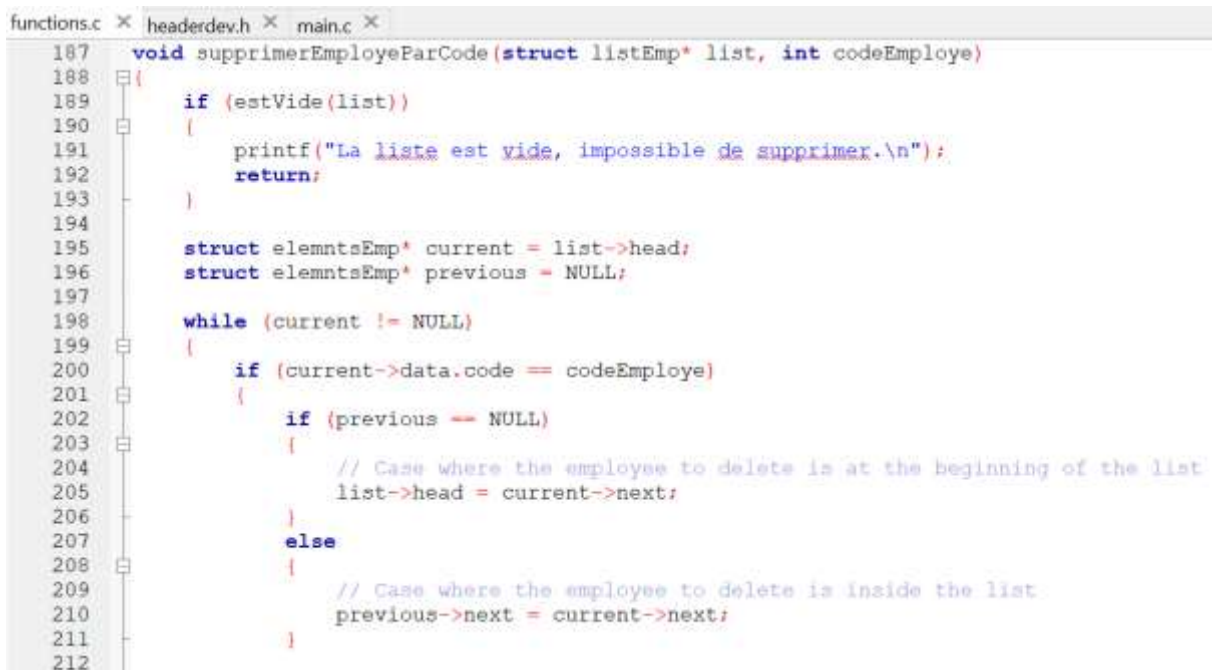
165
166 void ajouterEmploye(struct listEmp* list, struct employe emp)
167 {
168     struct elemntsEmp* nouveau = malloc(sizeof(struct elemntsEmp));
169     nouveau->data = emp;
170     nouveau->next = NULL;
171
172     if(list->head == NULL)
173     {
174         list->head = nouveau;
175     }
176     else
177     {
178         struct elemntsEmp* it = list->head;
179         while(it->next != NULL)
180         {
181             it = it->next;
182         }
183         it->next = nouveau;
184     }
185
186 }

```

Figure 3:Le code d'ajout d'un nouvel employé

b) Supprimer des Employés Existants

L'utilisateur peut sélectionner un employé à supprimer, et le programme affiche un message de confirmation avant de procéder à la suppression. Cela évite les suppressions involontaires.



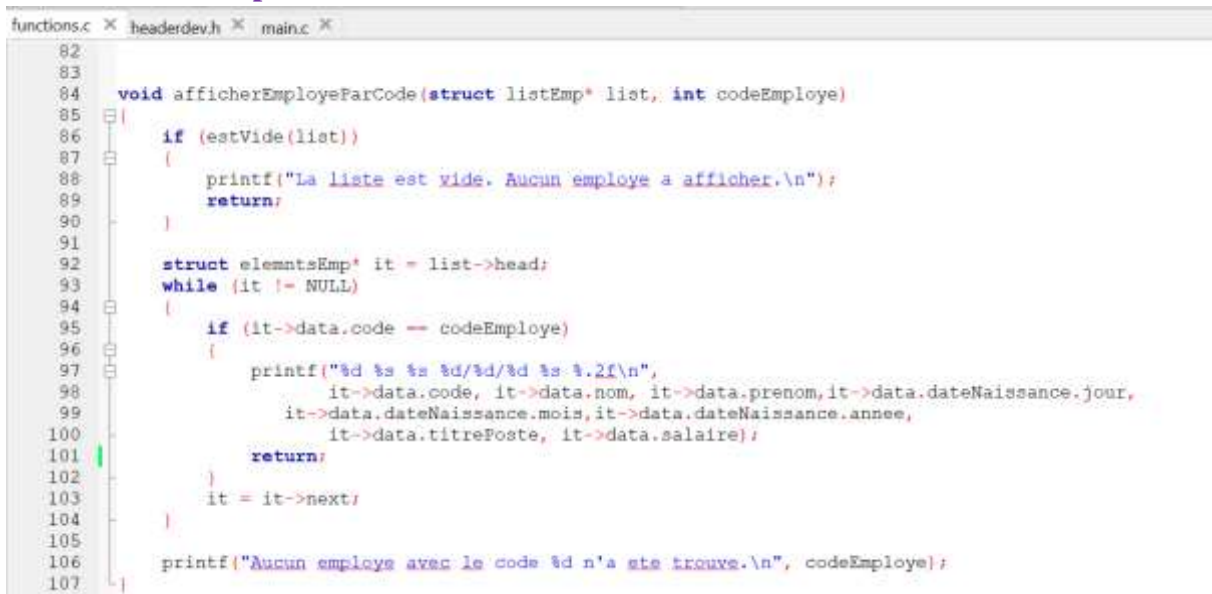
```
187 void supprimerEmployeParCode(struct listEmp* list, int codeEmploye)
188 {
189     if (estVide(list))
190     {
191         printf("La liste est vide, impossible de supprimer.\n");
192         return;
193     }
194
195     struct elemntsEmp* current = list->head;
196     struct elemntsEmp* previous = NULL;
197
198     while (current != NULL)
199     {
200         if (current->data.code == codeEmploye)
201         {
202             if (previous == NULL)
203             {
204                 // Case where the employee to delete is at the beginning of the list
205                 list->head = current->next;
206             }
207             else
208             {
209                 // Case where the employee to delete is inside the list
210                 previous->next = current->next;
211             }
212         }
213     }
214 }
```

Figure 4: Le code de suppression de l'employé

c) Rechercher un Employé

Le programme permet à l'utilisateur de rechercher un employé par code, nom ou prénom. La recherche par code garantit un résultat unique, tandis que la recherche par nom ou prénom peut retourner plusieurs résultats. En cas de recherche infructueuse, le programme informe l'utilisateur.

Recherche par code :



```
82
83
84 void afficherEmployeParCode(struct listEmp* list, int codeEmploye)
85 {
86     if (estVide(list))
87     {
88         printf("La liste est vide. Aucun employe a afficher.\n");
89         return;
90     }
91
92     struct elemntsEmp* it = list->head;
93     while (it != NULL)
94     {
95         if (it->data.code == codeEmploye)
96         {
97             printf("%d %s %s %d/%d/%d %s %.2f\n",
98                 it->data.code, it->data.nom, it->data.prenom, it->data.dateNaissance.jour,
99                 it->data.dateNaissance.mois, it->data.dateNaissance.annee,
100                 it->data.titrePoste, it->data.salaire);
101             return;
102         }
103         it = it->next;
104     }
105
106     printf("Aucun employe avec le code %d n'a été trouve.\n", codeEmploye);
107 }
```

Figure 5: Le code de rechercher l'employé par code

Rechercher par nom :

```
functions.c × headerdev.h × main.c ×
108
109 void afficherEmployeParNom(struct listEmp* list, char nomEmploye[])
110 {
111     if (estVide(list))
112     {
113         printf("La liste est vide. Aucun employe a afficher.\n");
114         return;
115     }
116     int found = 0;
117     struct elemntsEmp* it = list->head;
118     while (it != NULL)
119     {
120         if (strcmp(it->data.nom, nomEmploye) == 0)
121         {
122             printf("%d %s %s %d/%d/%d %s %.2f\n",
123                 it->data.code, it->data.nom, it->data.prenom, it->data.dateNaissance.jour,
124                 it->data.dateNaissance.mois, it->data.dateNaissance.annee,
125                 it->data.titrePoste, it->data.salaire);
126             found = 1;
127         }
128         it = it->next;
129     }
130     if(!found)
131     {
132         printf("Aucun employe avec le nom %s n'a ete trouve.\n", nomEmploye);
133     }
134 }
```

Figure 6: Le code de rechercher l'employé par nom

Rechercher par prénom :

```
functions.c × headerdev.h × main.c ×
135
136 void afficherEmployeParPrenom(struct listEmp* list, char prenomEmploye[])
137 {
138     if (estVide(list))
139     {
140         printf("La liste est vide. Aucun employe a afficher.\n");
141         return;
142     }
143     struct elemntsEmp* it = list->head;
144     int found = 0;
145     while (it != NULL)
146     {
147         if (strcmp(it->data.prenom, prenomEmploye) == 0)
148         {
149             printf("%d %s %s %d/%d/%d %s %.2f\n",
150                 it->data.code, it->data.nom, it->data.prenom, it->data.dateNaissance.jour,
151                 it->data.dateNaissance.mois, it->data.dateNaissance.annee,
152                 it->data.titrePoste, it->data.salaire);
153             found = 1;
154         }
155         it = it->next;
156     }
157     if(!found)
158     {
159         printf("Aucun employe avec le prenom %s n'a ete trouve.\n", prenomEmploye);
160     }
161 }
```

Figure 7: Le code de rechercher l'employé par prénom

d) Mettre à Jour les Informations d'un Employé

L'utilisateur peut mettre à jour les informations (poste et salaire) d'un employé en fournissant son code. Le programme limite les modifications aux champs spécifiés.


```

functions.c x headerdev.h x main.c x
351
352 void modificationEmp(struct listEmp* list, int codeEmploye)
353 {
354     int found = 0;
355     char res[10];
356     if (estVide(list))
357     {
358         printf("La liste est vide. Aucun employe a afficher.\n");
359         return;
360     }
361
362     struct elemntsEmp* it = list->head;
363     while (it != NULL)
364     {
365         if (it->data.code == codeEmploye)
366         {
367             afficherEmployeParCode(list, codeEmploye);
368             printf("qu'est ce vous voulez modifier ? ");
369             printf("-- Nom -- Prenom -- Poste -- Salaire -- DateNaiss : ");
370             scanf("%s", res);
371             if(strcmp(res, "Nom") == 0)
372             {
373                 saisieNom(list, &(it->data));
374                 printf("nom modifie avec succes \n");
375             }
376             if(strcmp(res, "Prenom") == 0)

```

Figure 8: Le code de modifier l'employé par code

e) Enregistrer les Informations dans le Fichier

L'utilisateur peut enregistrer les informations dans le fichier "employees.txt" à tout moment, même avant la fin du programme. Cela assure une sauvegarde régulière des données.

```

functions.c x headerdev.h x main.c x
402     printf("Aucun employe avec le code %d n'a ete trouve.\n", codeEmploye);
403 }
404 void sauvegarderListeDansFichier(struct listEmp* list, const char* fileName)
405 {
406     FILE* fichier = fopen(fileName, "w");
407
408     if (fichier == NULL)
409     {
410         printf("Erreur lors de l'ouverture du fichier %s\n", fileName);
411         return;
412     }
413
414     struct elemntsEmp* it = list->head;
415     while (it != NULL)
416     {
417         fprintf(fichier, "%d %s %s %d/%d/%d %s %.2f\n",
418             it->data.code, it->data.nom, it->data.prenom, it->data.dateNaissance.jour,
419             it->data.dateNaissance.mois, it->data.dateNaissance.annee,
420             it->data.titrePoste, it->data.salaire);
421         it = it->next;
422     }
423
424     fclose(fichier);
425     printf("Liste sauvegardee dans le fichier %s avec succes.\n", fileName);
426 }

```

Figure 9: Le code de sauvegarde

f) Quitter le Programme

Lorsque l'utilisateur choisit de quitter le programme (le choix 7) toutes les modifications apportées pendant l'exécution sont implicitement enregistrées.

Partie Console :

Menu de choix :

```
C:\Users\lenovo\Desktop\Devoir2-ENS-TMW-master\main.exe
liste chargee depuis le fichier employees.txt avec succes.
      Bienvenu
-----
Veuillez choisir une des operations suivantes:
-----
1. Afficher la liste des employes
2. Ajouter un nouveau employe
3. Supprimer un employe
   1. Supprimer un employe avec son code
   2. Supprimer un employe avec son nom
4. Rechercher un employe par son code
   1. Rechercher un employe par son code
   2. Rechercher un employe son nom
   3. Rechercher un employe son prenom
5. Modification
6. afficher Menu
7. Quitter
-----
veuillez effectuer votre choix : _
```

Figure 10: Le menu d'affichage

L'opération d'affichage des employés :

```
C:\Users\lenovo\Desktop\Devoir2-ENS-TMW-master\main.exe
veuillez effectuer votre choix : 1
-----
voici la liste des employees :
1001 Murphy Diane 5/1/1970 President 150000.00
1056 Patterson Mary 2/3/1981 VP-Sales 100000.00
1076 Firrelli Jeff 20/12/1983 VP-Marketing 90000.00
1088 Patterson William 30/4/1978 Sales-Manager 50000.00
1102 Bondurand Gerard 28/7/1976 Sales-Manager 52000.00
1143 Bow Anthony 13/4/1977 Sales-Manager 50000.00
1165 Jennings Leslie 21/1/1985 Sales-Rep 15000.00
1166 Thompson Mark 15/2/1983 Sales-Rep 25000.00
1188 Firrelli Julie 19/8/1990 Sales-Rep 12000.00
1216 Patterson Steve 30/9/1986 Sales-Rep 12800.00
1286 Tseng Foon 18/3/1976 Sales-Rep 27400.00
1323 Vanauf George 2/2/1996 Sales-Rep 10000.00
1337 Carpenter Louis 10/11/2000 Sales-Rep 9000.00
1370 Hernandez Gerard 23/9/1990 Sales-Rep 12600.00
1401 Castillo Pamela 4/6/1977 Sales-Rep 32000.00
1501 Bott Larry 20/12/1989 Sales-Rep 15900.00
1504 Jones Barry 30/5/1998 Sales-Rep 12900.00
1611 Fixter Andy 4/4/2001 Sales-Rep 12200.00
1612 Marsh Peter 10/3/1984 Sales-Rep 23000.00
1619 King Tom 1/1/2003 Sales-Rep 9000.00
1621 Nishis Marie 17/8/1986 Sales-Rep 21000.00
1625 Kato Yoshimi 29/1/1987 Sales-Rep 32000.00
-----
veuillez effectuer votre choix : _
```

Figure 11: La liste d'employés

L'opération d'ajout d'un employé :

```
veuillez effectuer votre choix : 2
-----
pour ajouter un nouveau employe veuillez entrer les informations suivantes :
Code : 0000
Nom : Hachimi Moueddene
Prenom : date naissance :
    jour : 09
    mois : 05
    annee : 2004
Salaire : 2345.00
les postes qui existent sont :
-- President -- VP-Sales -- VP-Marketing -- Sales-Manager -- Sales-Rep
Titre du poste : President
employe ajoute avec succes
-----
veuillez effectuer votre choix : 
```

Figure 12: L'ajout d'un nouvel employé

L'opération de suppression par code :

```
-----
veuillez effectuer votre choix : 3
-----
pour supprimer un employe veuillez effectuer votre choix de suppression : 1
entrer le code de l'employe que vous voulez supprimer : 1625
1625 Kato Yoshimi 29/1/1987 Sales-Rep 32000.00
Voulez-vous vraiment supprimer cet employe ? [yes | no] : yes
Employe avec le code 1625 supprime avec succes.
-----
veuillez effectuer votre choix :
```

Figure 13: La suppression par code

Conclusion

En utilisant le langage C, ce programme offre une solution complète et efficace pour la gestion d'une liste d'employés. Les structures de données bien définies et la gestion judicieuse des opérations de lecture/écriture garantissent la robustesse et la performance du système.