

Langage procédural : Listes chaînées

# Gestion de restaurant

Réalisé par

AUVERSACK DAMIEN

HOUDART FLORAN

Année académique 2020–2021

## Table des matières

|  |    |
|--|----|
| Langage procédural : Listes chaînées ..... | 1  |
| Table des matières .....                   | 2  |
| Notion de liste chaînée .....              | 3  |
| Enoncé personnalisé .....                  | 4  |
| Mode d'emplois de l'application .....      | 5  |
| Explication globale de l'application ..... | 9  |
| Liste des structures utilisées .....       | 12 |
| Programme .....                            | 13 |
| AuversackHoudart_Main.c .....              | 13 |
| AuversackHoudart_affichage.c .....         | 16 |
| AuversackHoudart_ajout.c .....             | 21 |
| AuversackHoudart_compter.c .....           | 25 |
| AuversackHoudart_operation.c .....         | 28 |
| AuversackHoudart_outils.c .....            | 34 |
| AuversackHoudart_recuperation.c .....      | 44 |
| AuversackHoudart_suppression.c .....       | 53 |
| AuversackHoudart_affichage.h .....         | 57 |
| AuversackHoudart_ajout.h .....             | 58 |
| AuversackHoudart_compter.h .....           | 59 |
| AuversackHoudart_operation.h .....         | 60 |
| AuversackHoudart_outils.h .....            | 61 |
| AuversackHoudart_recuperation.h .....      | 62 |
| AuversackHoudart_suppression.h .....       | 63 |
| AuversackHoudart_structures.h .....        | 64 |
| AuversackHoudart_ressources.rc .....       | 65 |

## Notion de liste chaînée

Une liste chaînée c'est une liste de structures composée de n'importe quel type de variable mais contenant un pointeur du type de la structure et pointant vers l'adresse de l'élément suivant de la liste.

Dans une liste chaînée, la taille est allouée dynamiquement grâce aux pointeurs en utilisant la fonction « *malloc()* » (memory allocation) et prenant comme argument la taille nécessaire. Pour cela nous utilisons la fonction « *sizeof()* » avec comme argument le nom de notre structure.

Le dernier élément d'une liste chaînée contiendra toujours dans son élément suivant la valeur NULL

Lorsqu'un élément alloué avec « *malloc()* » n'est plus utilisé, son espace occupé en mémoire peut être libéré avec la fonction « *free()* »

Pour travailler avec une liste chaînée et ajouter, supprimer et intercaler des éléments dans la liste, nous créons quatre pointeurs différents du type de la structure que nous nommons « *deb* », « *courant* », « *suivant* » et « *intercale* ».

« *deb* » contiendra l'adresse du premier élément de la liste chaînée.

« *courant* » contiendra l'élément courant de la liste chaînée que va varier en fonction de l'emplacement du pointeur lors du parcourt de la liste. Il prendra l'adresse de l'élément suivant lors de la prochaine itération.

« *suivant* » contiendra l'adresse de l'élément suivant de l'élément courant et donnera son adresse au pointeur courant à la prochaine itération.

« *intercale* » permettra d'insérer un élément dans la liste à n'importe quelle position. Pour cela l'élément suivant contenu dans intercale va prendre comme valeur l'adresse de l'élément qui suit celui qui doit être inséré. L'élément précédent celui qui doit être inséré prendra comme valeur dans son élément suivant l'adresse de l'élément inséré.

Si l'élément est inséré en première position, alors l'élément suivant de l'élément inséré prendra l'adresse de *deb* et *deb* prendra l'adresse de *intercale*. Si l'élément inséré est en dernière position, alors l'élément suivant du dernier élément de la liste chaînée aura comme valeur l'adresse d'*intercale* et l'élément suivant d'*intercale* prendra la valeur NULL.

## Enoncé personnalisé

Le patron d'un restaurant propose différents menus, dispose d'un certain nombre de tables, et à chaque table on peut placer un certain nombre maximum de personnes. Il est possible d'effectuer une réservation de la table et du menu (un menu par table), on peut aussi modifier son menu sur place. On peut faire une réservation sur place, il faut alors s'assurer qu'il reste de la place dans le restaurant avant d'accepter le client.

Le patron gère aussi les services (midi ou soir) et la fonction de chaque employé. L'addition est calculée pour une table en fonction de son menu et du nombre de personnes à table. Le patron doit pouvoir moduler son restaurant en changeant le nombre de tables, de menus et d'employés.

## Mode d'emplois de l'application

L'application conçue permet de gérer un restaurant. Pour cela un programme nommé AuversackHoudart.exe doit être ouvert pour accéder au logiciel.

Le logiciel possède un menu principal qui donne le choix à l'utilisateur entre la gestion du personnel (ajouter, supprimer ou modifier un employé) et les services proposés par le restaurant (réservation, commande, addition, gestion des tables et gestion des menus).

Pour naviguer dans les menus il suffira toujours d'entrer le numéro du menu que l'on souhaite ouvrir. Pour exemple, rentrer « 1 » dans le menu de gestion du restaurant ouvrira le menu de gestion du personnel.

Pour revenir dans le menu précédent il faut entrer « 0 » lorsque l'option est disponible. Si le « 0 » n'est pas visible, alors il faut terminer l'opération en cours.

Le programme utilise des fichiers de données pour sauvegarder les informations entrées et ainsi les réutiliser à la prochaine ouverture.

Voici ci-dessous le menu de gestion du restaurant, le menu de gestion du personnel et le menu des services qui sont les trois menus principaux.

### Le menu de gestion du restaurant :



### Le menu de gestion du personnel :



The screenshot shows a terminal window titled "Gestion Restaurant". The main menu is "Gestion du personnel" displayed in a large, stylized, outlined font. Below the title, there are two lines of asterisks. The service hours are listed as "Service - Midi : 12h00-15h00 - Soir : 18h00-23h00". Below this, the prompt "|Employe|" is shown. A table header is displayed with columns: "N", "Nom", "Service", and "Fonction". Below the header, there is a list of options: "1. Ajouter un employe", "2. Supprimer un employe", "3. Modifier un employe", and "0. Retour". At the bottom, the prompt "Choix :" is shown.

```

*****
Gestion du personnel
*****

Service - Midi : 12h00-15h00 - Soir : 18h00-23h00

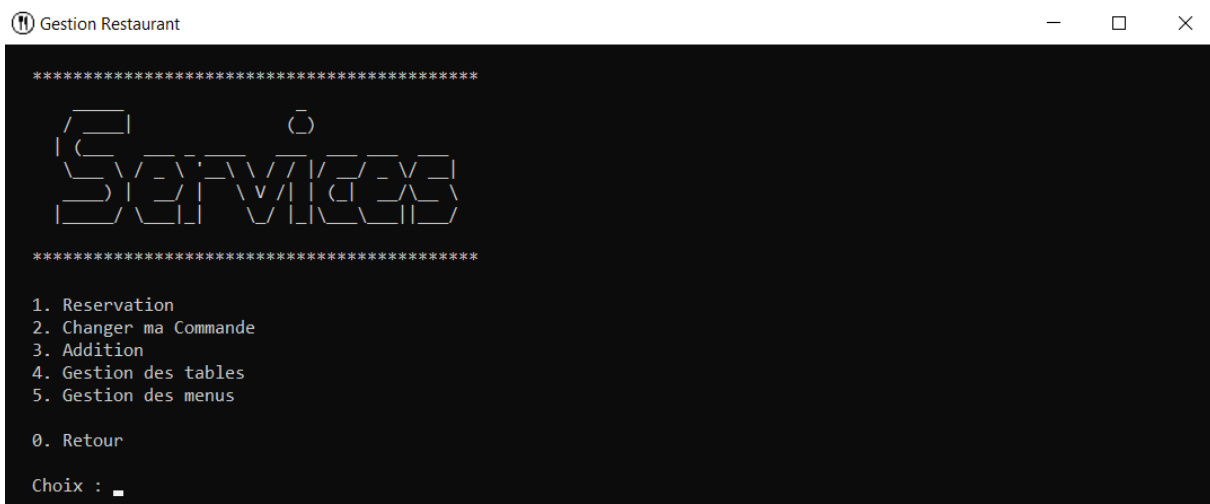
|Employe|
-----
| N |   Nom   | Service | Fonction |
-----

1. Ajouter un employe
2. Supprimer un employe
3. Modifier un employe

0. Retour

Choix :
  
```

### Le menu des services :



The screenshot shows a terminal window titled "Gestion Restaurant". The main menu is "Services" displayed in a large, stylized, outlined font. Below the title, there are two lines of asterisks. The list of options is: "1. Reservation", "2. Changer ma Commande", "3. Addition", "4. Gestion des tables", "5. Gestion des menus", and "0. Retour". At the bottom, the prompt "Choix : " is shown with a cursor.

```

*****
Services
*****

1. Reservation
2. Changer ma Commande
3. Addition
4. Gestion des tables
5. Gestion des menus

0. Retour

Choix : 
  
```

Chaque menu a une liste d'actions possibles. Voici une liste exhaustive de toutes les actions possibles en fonction du menu :

**A. Gestion restaurant**

**a. Gestion du personnel**

i. Ajouter un employé

Permet d'ajouter un employé avec son nom, sa fonction et son service.

ii. Supprimer un employé

Permet de supprimer un employé par son numéro.

iii. Modifier un employé

1. *Modifier le nom*

Permet de modifier le nom d'un employé.

2. *Modifier le service*

Permet de modifier le service d'un employé (midi ou soir).

3. *Modifier la fonction*

Permet de modifier la fonction d'un employé au sein du restaurant.

**b. Services**

i. Réservation

1. *Faire une réservation*

Il faut entrer les différentes informations nécessaires, comme le nom, le nombre de personnes, le service et le menu dans l'ordre demander par le programme.

2. *Supprimer une réservation*

Il faut entrer le service et le numéro de table pour supprimer la réservation.

ii. Changer ma commande

Il faut entrer le service, le numéro de la table et ensuite le numéro du nouveau menu.

iii. Addition

1. *Faire l'addition (service du midi)*

Permet de calculer l'addition d'une table pour le service du midi.

2. *Faire l'addition (service du soir)*

Permet de calculer l'addition d'une table pour le service du soir.

iv. Gestion des tables

1. *Ajouter une table*

Il faut entrer le nombre de place maximum de la table, on peut aussi déjà réserver la table à sa création ou non pour chaque service.

2. *Supprimer une table*

Il faut entrer le numéro de la table pour la supprimer, cela supprimera aussi toutes les réservations sur la table.

v. Gestion des menus

1. *Ajouter un menu*

Pour ajouter un menu, il faut son nom, son prix et sa description.

2. *Supprimer un menu*

Pour supprimer un menu il faut simplement donner le numéro du menu pour le supprimer.



## Explication globale de l'application

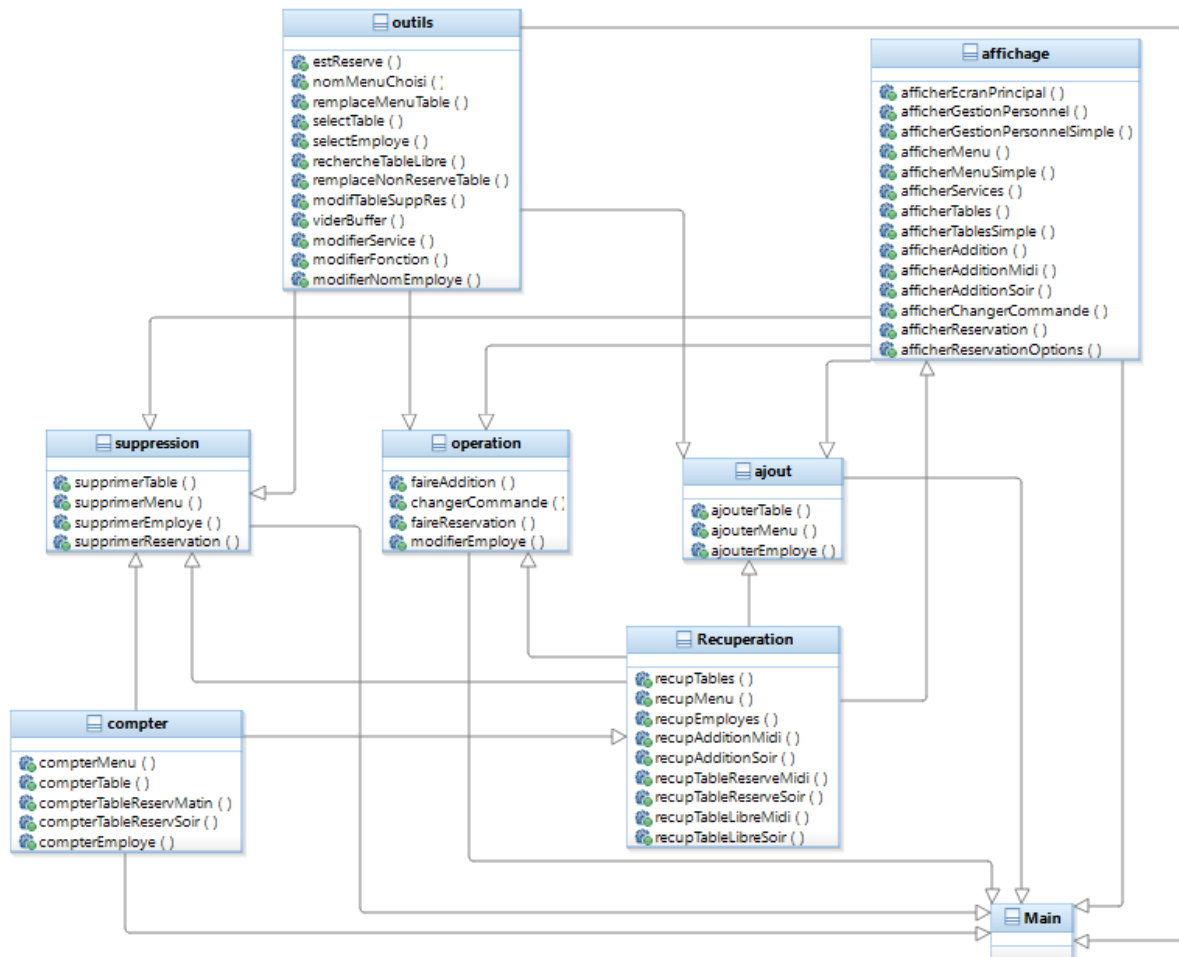
L'application est divisée en plusieurs fichiers qui contiennent chacun différents types de fonctions. Le fichier contenant le main utilise ces différentes fonctions pour le bon déroulement du programme. Ci-dessous vous trouverez deux schémas, le premier représente les différents fichiers, les fonctions qu'ils contiennent ainsi que les appels de fonctions entre les différents fichiers. Le second schéma contient l'arborescence des fonctions utilisées en partant du main.

L'application utilise trois fichiers de données pour gérer les employés, les tables et les menus. Les fichiers sont nommés « AuversackHoudart\_Employes.dat », « AuversackHoudart\_Table.dat » et « AuversackHoudart\_Menu.dat ». Ces fichiers sont mis-à-jour avec chaque opération, ainsi si le programme est fermé on conserve les données pour la prochaine utilisation.

Liste des fichiers ainsi que leurs descriptions :

- *AuversackHoudart\_main* : source du programme, contient l'exécution principale.
- *AuversackHoudart\_affichage* : gère les affichages en ASCII art de chaque menu
- *AuversackHoudart\_ajout* : gère l'ajout des tables, menus et employés dans les fichiers.
- *AuversackHoudart\_compter* : permet de compter le nombre de menus, tables (selon la réservation ou non), employés.
- *AuversackHoudart\_operation* : ensemble des opérations permettant le bon déroulement du programme.
- *AuversackHoudart\_outils* : ensemble des outils aidant à la manipulation des données.
- *AuversackHoudart\_recuperation* : permet de récupérer la liste des tables (réservées ou non, au matin ou au soir), menus et employés ainsi que de l'addition et de les afficher.
- *AuversackHoudart\_Suppression* : gère la suppression des tables, menus et employés dans les fichiers.

Les différents fichiers :



L'arborescence des fonctions :



## Liste des structures utilisées

### **Structure des tables :**

```
typedef struct Table {  
    int nbPlaceMax;  
    int estReserveMatin;  
    char nomMatin[21];  
    int nbPersonneMatin;  
    int numMenuMatin;  
    int estReserveSoir;  
    char nomSoir[21];  
    int nbPersonneSoir;  
    int numMenuSoir;  
    struct Table *suivant;  
}Table;
```

### **Structure des menus :**

```
typedef struct Menu {  
    char nom[21];  
    float prix;  
    char description[60];  
    struct Menu *suivant;  
}Menu;
```

### **Structure des employés :**

```
typedef struct Employe {  
    char nom[21];  
    int service;  
    char fonction[21];  
    struct Employe *suivant;  
}Employe;
```

## Programme

### AuversackHoudart\_Main.c

```

1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <string.h>
4  #include "Headers/AuversackHoudart_affichage.h"
5  #include "Headers/AuversackHoudart_ajout.h"
6  #include "Headers/AuversackHoudart_suppression.h"
7  #include "Headers/AuversackHoudart_recuperation.h"
8  #include "Headers/AuversackHoudart_operation.h"
9  #include "Headers/AuversackHoudart_compter.h"
10 main() {
11     // Paramètre Fenêtre
12     system("title Gestion Restaurant"); // Définit un titre à ma fenêtre
13     system("mode con: cols=120 lines=50"); // Définit la taille de ma fenêtre
14     // Fin Paramètre Fenêtre
15     int choix=0, choixGestionPersonnel=0, choixServices=0, choixMenu=0,
16     choixTables=0, choixAddition=0, tableAddition=0, choixReservation=0, testDigit;
17     while(1) { // Ecran Principal
18         erreurEcranPrincipal://gestion d'erreur
19         system("cls");
20         afficherEcranPrincipal();
21         testDigit = scanf("%d", &choix);
22         if(testDigit==0) {
23             viderBuffer();
24             goto erreurEcranPrincipal;
25         }
26         switch(choix) {
27             case 1: // Gestion du personnel
28                 while(1) {
29                     erreurGestionPersonnel://gestion d'erreur
30                     system("cls");
31                     afficherGestionPersonnel();
32                     testDigit = scanf("%d", &choixGestionPersonnel);
33                     if(testDigit==0) {
34                         viderBuffer();
35                         goto erreurGestionPersonnel;
36                     }
37                     switch(choixGestionPersonnel) {
38                         case 1:
39                             system("cls");
40                             ajouterEmploye();
41                             break;
42                         case 2:
43                             system("cls");
44                             supprimerEmploye();
45                             break;
46                         case 3:
47                             system("cls");
48                             if(compterEmploye()!=0) {
49                                 modifierEmploye();
50                                 system("pause");
51                             }
52                             break;
53                     }
54                     if(choixGestionPersonnel==0) break;
55                 }
56                 break;
57             case 2: // Services
58                 while(1) {
59                     erreurServices://gestion d'erreur
60                     system("cls");
61                     afficherServices();
62                     testDigit = scanf("%d", &choixServices);
63                     if(testDigit==0) {
64                         viderBuffer();
65                         goto erreurServices;
66                     }
67                     switch(choixServices) {
68                         case 1: // Services -> 1. Faire une reservation
69                             while(1) {
70                                 erreurReservationOptions://gestion d'erreur
71                                 system("cls");
72                                 afficherReservationOptions();

```

```

73     testDigit = scanf("%d", &choixReservation);
74     if(testDigit==0) {
75         viderBuffer();
76         goto erreurReservationOptions;
77     }
78     switch(choixReservation) {
79         case 1:
80             system("cls");
81             faireReservation();
82             system("pause");
83             break;
84         case 2:
85             system("cls");
86             if(compterTableReservMatin()!=0 ||
87                 compterTableReservSoir()!=0) {
88                 supprimerReservation();
89                 system("pause");
90             }
91             break;
92     }
93     if(choixReservation==0)
94         break;
95
96     case 2: // Services -> 2. Changer sa commande
97         system("cls");
98         if(compterTableReservSoir()!=0 ||
99             compterTableReservMatin()!=0) {
100             changerCommande();
101             system("pause");
102         }
103         break;
104
105     case 3: // Services -> 3. Addition
106         while(1) {
107             erreurChoixAddition://gestion d'erreur
108             system("cls");
109             afficherAddition();
110             testDigit = scanf("%d", &choixAddition);
111             if(testDigit==0) {
112                 viderBuffer();
113                 goto erreurChoixAddition;
114             }
115             switch(choixAddition) {
116                 case 1:
117                     erreurAdditionMidi://gestion
118                     d'erreur
119                     system("cls");
120                     afficherAdditionMidi();
121                     testDigit = scanf("%d", &tableAddition);
122                     if(testDigit==0) {
123                         viderBuffer();
124                         goto erreurAdditionMidi;
125                     }
126                     FaireAddition(tableAddition, 1);
127                     system("pause");
128                     break;
129                 case 2:
130                     erreurAdditionSoir://gestion
131                     d'erreur
132                     system("cls");
133                     afficherAdditionSoir();
134                     testDigit = scanf("%d", &tableAddition);
135                     if(testDigit==0) {
136                         viderBuffer();
137                         goto erreurAdditionSoir;
138                     }
139                     FaireAddition(tableAddition, 2);
140                     system("pause");
141                     break;
142             }
143         }

```

```

134         FaireAddition(tableAddition, 2);
135         system("pause");
136         break;
137     }
138     if(choixAddition==0) break;
139 }
140 break;
141 case 4: // Services -> 4. Gestion des tables
142 while(1) {
143     erreurAfficherTables://gestion d'erreur
144     system("cls");
145     afficherTables();
146     testDigit = scanf("%d", &choixTables);
147     if(testDigit==0) {
148         viderBuffer();
149         goto erreurAfficherTables;
150     }
151     switch(choixTables) {
152         case 1: // ajouter Table
153             system("cls");
154             if(compterMenu()!=0) {
155                 ajouterTable();
156             }
157             break;
158         case 2: // supprimer Table
159             system("cls");
160             if(compterTable()!=0) {
161                 supprimerTable();
162             }
163             break;
164     }
165     if(choixTables==0) break;
166 }
167 break;
168 case 5: // Services -> 5. Gestion du menu
169 while(1) {
170     erreurAfficherMenu://gestion d'erreur
171     system("cls");
172     afficherMenu();
173     testDigit = scanf("%d", &choixMenu);
174     if(testDigit==0) {
175         viderBuffer();
176         goto erreurAfficherMenu;
177     }
178     switch(choixMenu) {
179         case 1: // ajouter Menu
180             system("cls");
181             ajouterMenu();
182             break;
183         case 2: // supprimer Menu
184             system("cls");
185             supprimerMenu();
186             break;
187     }
188     if(choixMenu==0) break;
189 }
190 break;
191 } //retour au menu principal
192 if(choixServices==0) break;
193
194 } //sortie du menu de gestion
195 break;
196 }
197 if(choix==0) break;
198 }

```



# AuversackHoudart\_affichage.c

```

1  #include <stdio.h>
2  #include <stdlib.h>
3  void afficherEcranPrincipel() { //gestion restaurant + options
4      printf("\n");
5      printf("
6      .....
7      .....");
8      printf("\n");
9      printf("
10     .....");
11     printf("\n");
12     printf("
13     .....");
14     printf("\n");
15     printf("
16     .....");
17     printf(" 1. Gestion du personnel\n");
18     printf(" 2. Services\n");
19     printf("\n");
20     printf(" 0. Quitter\n");
21     printf("\n");
22     printf(" Choix : ");
23 }
24 void afficherGestionPersonnel() { //gestion du personnel + options
25     printf("\n");
26     printf("
27     .....");
28     printf("\n");
29     printf("
30     .....");
31     printf("\n");
32     printf("
33     .....");
34     printf("\n");
35     printf("
36     .....");
37     printf("\n");
38     printf("
39     .....");
40     printf(" 1. Ajouter un employe\n");
41     printf(" 2. Supprimer un employe\n");
42     printf(" 3. Modifier un employe\n");
43     printf("\n");
44     printf(" 0. Retour\n");
45     printf("\n");
46     printf(" Choix : ");
47 }
48 void afficherGestionPersonnelSimple() { //gestion du personnel + options
49     printf("\n");
50     printf("
51     .....");

```



```

51     .....");
52     printf("\n");
53     printf("      /____|      | | | |      | \n");
54     printf("      | | | |      | | | |      | \n");
55     printf("      | | | |      | | | |      | \n");
56     printf("      | | | |      | | | |      | \n");
57     printf("      | | | |      | | | |      | \n");
58     printf("      | | | |      | | | |      | \n");
59     printf("      | | | |      | | | |      | \n");
60     printf("\n");
61     printf(".....");
62     printf("\n");
63 }
64 void afficherMenu() { //gestion des menus + options
65     printf("\n");
66     printf(".....");
67     printf("\n");
68     printf("      /____|      | | | |      | \n");
69     printf("      | | | |      | | | |      | \n");
70     printf("      | | | |      | | | |      | \n");
71     printf("      | | | |      | | | |      | \n");
72     printf("      | | | |      | | | |      | \n");
73     printf("      | | | |      | | | |      | \n");
74     printf("\n");
75     printf(".....");
76     printf("\n");
77     recupMenu();
78     printf(" 1. Ajouter un menu\n");
79     printf(" 2. Supprimer un menu\n");
80     printf("\n");
81     printf(" 0. Retour\n");
82     printf("\n");
83     printf(" Choix : ");
84 }
85 void afficherMenuSimple() { //gestion des menus
86     printf("\n");
87     printf(".....");
88     printf("\n");
89     printf("      /____|      | | | |      | \n");
90     printf("      | | | |      | | | |      | \n");
91     printf("      | | | |      | | | |      | \n");
92     printf("      | | | |      | | | |      | \n");
93     printf("      | | | |      | | | |      | \n");
94     printf("      | | | |      | | | |      | \n");

```

```

95     printf("\n");
96     printf("
.....
.....");
97     printf("\n");
98 }
99 void afficherServices() { //services + options
100     printf("\n");
101     printf("
.....");
102     printf("\n");
103     printf("
          \n");
104     printf("
          \n");
105     printf("
          \n");
106     printf("
          \n");
107     printf("
          \n");
108     printf("
          \n");
109     printf("\n");
110     printf("
.....");
111     printf("\n");
112     printf("\n");
113     printf(" 1. Reservation\n");
114     printf(" 2. Changer sa Commande\n");
115     printf(" 3. Addition\n");
116     printf(" 4. Gestion des tables\n");
117     printf(" 5. Gestion des menus\n");
118     printf("\n");
119     printf(" 0. Retour\n");
120     printf("\n");
121     printf(" Choix : ");
122 }
123 void afficherTables() { //gestion des tables + options
124     printf("\n");
125     printf("
.....
.....");
126     printf("\n");
127     printf("
          \n");
128     printf("
          \n");
129     printf("
          \n");
130     printf("
          \n");
131     printf("
          \n");
132     printf("
          \n");
133     printf("\n");
134     printf("
.....
.....");
135     printf("\n");
136     recupTables();
137     printf(" 1. Ajouter une table\n");
138     printf(" 2. Supprimer une table\n");
139     printf("\n");
140     printf(" 0. Retour\n");
141     printf("\n");
142     printf(" Choix : ");
143 }
144 void afficherTablesSimple() { //gestion des tables + options
145     printf("\n");
146     printf("
.....
.....");
147     printf("\n");
148     printf("
          \n");
149     printf("
          \n");
150     printf("
          \n");

```

```
216     printf("\n");
```

```

217     printf("
218     printf("
219     printf("
220     printf("
221     printf("
222     printf("
223     printf("
224     printf("
225     printf("\n");
226     printf("
227     printf("\n");
228 }
229 void afficherReservation() { //reservation
230     printf("\n");
231     printf("
232     printf("\n");
233     printf("
234     printf("
235     printf("
236     printf("
237     printf("
238     printf("
239     printf("\n");
240     printf("
241     printf("\n");
242 }
243 void afficherReservationOptions() { //reservation + options
244     printf("\n");
245     printf("
246     printf("\n");
247     printf("
248     printf("
249     printf("
250     printf("
251     printf("
252     printf("
253     printf("\n");
254     printf("
255     printf("\n");
256     printf("\n");
257     printf(" 1. Faire une reservation\n");
258     printf(" 2. Supprimer une reservation\n");
259     printf("\n");
260     printf(" 0. Retour\n");
261     printf("\n");
262     printf(" Choix : ");
263 }

```

# AuversackHoudart\_ajout.c

```

1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <string.h>
4  #include "../Headers/AuversackHoudart_structures.h"
5  void ajouterTable() { //permet d'ajouter une table a la liste
6      int testDigit=0, j;
7      Table table;
8      FILE *fdat;
9      fdat = fopen("Data/AuversackHoudart_Table.dat", "a");
10     erreurIndiceNbPlaceMax: //gestion d'erreur
11     // Ajout table
12     system("cls");
13     afficherTablesSimple();
14     recupTables();
15     printf("  Nombre de places a table : ");
16     testDigit = scanf("%d", &table.nbPlaceMax);
17     if(table.nbPlaceMax<=0 || testDigit==0) {
18         viderBuffer();
19         goto erreurIndiceNbPlaceMax;
20     }
21     erreurEstReserveMatin: //gestion d'erreur
22     system("cls");
23     afficherTablesSimple();
24     recupTables();
25     printf("  Table reserve le matin(sui=1/non=0) : ");
26     testDigit = scanf("%d", &table.estReserveMatin);
27     if( (table.estReserveMatin != 0 && table.estReserveMatin != 1) || testDigit==0 ) {
28         viderBuffer();
29         goto erreurEstReserveMatin;
30     }
31     if(table.estReserveMatin==1) {
32         system("cls");
33         afficherTablesSimple();
34         recupTables();
35         printf("  Nom de la reservation : ");
36         viderBuffer();
37         fgets(table.nonMatin, 10, stdin);
38         // remplace les ' ' par des '_' dans la chaine de caracteres
39         for(j=0; j<strlen(table.nonMatin); j++) {
40             if(table.nonMatin[j] == ' ') {
41                 table.nonMatin[j] = '_';
42             }
43             if(table.nonMatin[j] == '\n') {
44                 table.nonMatin[j] = '\0';
45             }
46         }
47         erreurNbPlaceMaxMatin: //gestion d'erreur
48         system("cls");
49         afficherTablesSimple();
50         recupTables();
51         printf("  Nombre de personne a table : ");
52         testDigit = scanf("%d", &table.nbPersonneMatin);
53         if(table.nbPersonneMatin > table.nbPlaceMax || table.nbPersonneMatin<=0 ||
54             testDigit==0) {
55             viderBuffer();
56             goto erreurNbPlaceMaxMatin;
57         }
58         erreurNumMenuMatin: //gestion d'erreur
59         system("cls");
60         afficherTablesSimple();
61         recupTables();
62         printf("  Numéro du menu choisi : ");
63         testDigit = scanf("%d", &table.numMenuMatin);
64         if(table.numMenuMatin<=0 || table.numMenuMatin>compterMenu() ||
65             testDigit==0) {
66             viderBuffer();
67             goto erreurNumMenuMatin;
68         }
69     }
70     erreurEstReserveSoir: //gestion d'erreur
71     system("cls");
72     afficherTablesSimple();
73     recupTables();

```



```

72 printf(" Table reserve le soir(cui=1/non=0) : ");
73 testDigit = scanf("%d", &table.estReserveSoir);
74 if( (table.estReserveSoir != 0 && table.estReserveSoir != 1) || testDigit==0 ) {
75     viderBuffer();
76     goto erreurEstReserveSoir;
77 }
78 if(table.estReserveSoir==1) {
79     system("cls");
80     afficherTablesSimple();
81     recupTables();
82     printf(" Nom de la reservation : ");
83     viderBuffer();
84     fgets(table.nomSoir, 12, stdin);
85     // remplace les ' ' par des '_' dans la chaine de caracteres
86     for(j=0; j<strlen(table.nomSoir); j++) {
87         if(table.nomSoir[j] == ' ') {
88             table.nomSoir[j] = '_';
89         }
90         if(table.nomSoir[j] == '\n') {
91             table.nomSoir[j] = '\0';
92         }
93     }
94     erreurNbPlaceMaxSoir: //gestion d'erreur
95     system("cls");
96     afficherTablesSimple();
97     recupTables();
98     printf(" Nombre de personne a table : ");
99     testDigit = scanf("%d", &table.nbPersonneSoir);
100     if(table.nbPersonneSoir > table.nbPlaceMax || table.nbPersonneSoir<=0 ||
101     testDigit==0) {
102         viderBuffer();
103         goto erreurNbPlaceMaxSoir;
104     }
105     erreurNumMenuSoir: //gestion d'erreur
106     system("cls");
107     afficherTablesSimple();
108     recupTables();
109     printf(" Numero du menu choisi : ");
110     testDigit = scanf("%d", &table.numMenuSoir);
111     if(table.numMenuSoir<=0 || table.numMenuSoir>compterMenu() || testDigit==0) {
112         viderBuffer();
113         goto erreurNumMenuSoir;
114     }
115     if(compterTable()!=0) {
116         fprintf(fdat, "\n");
117     }
118     fprintf(fdat, "%d ", table.estReserveMatin);
119     if(table.estReserveMatin==1) {
120         table.nbPersonneSoir = table.numMenuSoir = 0;
121         fprintf(fdat, "%12s %d %d ", table.nomMatin, table.nbPersonneMatin,
122             table.numMenuMatin);
123     }
124     fprintf(fdat, "%d", table.estReserveSoir);
125     if(table.estReserveSoir==1) {
126         table.nbPersonneMatin = table.numMenuMatin = 0;
127         fprintf(fdat, " %12s %d %d", table.nomSoir, table.nbPersonneSoir,
128             table.numMenuSoir);
129     }
130     fprintf(fdat, " %d", table.nbPlaceMax);
131     fclose(fdat);
132 }
133 void ajouterMenu() { //permet d'ajouter un menu
134     int j;
135     Menu menu;
136     FILE *fdat;
137     fdat = fopen("Data/AuversackHoudart_Menu.dat", "a");
138     afficherMenuSimple();
139     recupMenu();
140     printf(" Nom du Menu : ");
141     viderBuffer();
142     fgets(menu.nom, 20, stdin);
143     // remplace les ' ' par des '_' dans la chaine de caracteres

```

```

142     for(j=0; j<strlen(menu.nom); j++) {
143         if(menu.nom[j] == ' ') {
144             menu.nom[j] = '_';
145         }
146         if(menu.nom[j] == '\n') {
147             menu.nom[j] = '\0';
148         }
149     }
150     system("cls");
151     afficherMenuSimple();
152     recupMenu();
153     printf(" Prix du Menu : ");
154     scanf("%f", &menu.prix);
155     system("cls");
156     afficherMenuSimple();
157     recupMenu();
158     printf(" Description du Menu : ");
159     viderBuffer();
160     fgets(menu.description, 55, stdin);
161     // remplace les ' ' par des '_' dans la chaine de caracteres
162     for(j=0; j<strlen(menu.description); j++) {
163         if(menu.description[j] == ' ') {
164             menu.description[j] = '_';
165         }
166         if(menu.description[j] == '\n') {
167             menu.description[j] = '\0';
168         }
169     }
170     if(compterMenu()!=0) {
171         fprintf(fdat, "\n");
172     }
173     fprintf(fdat, "%s %5.2f %s", menu.nom, menu.prix, menu.description);
174     fclose(fdat);
175 }
176 void ajouterEmploye() { //permet d'ajouter un employé à la liste
177     int j, testDigit;
178     Employe employe;
179     FILE *fdat;
180     fdat = fopen("Data/AnversackHoudart_Employes.dat", "a");
181     afficherGestionPersonnelSimple();
182     recupEmployes();
183     printf(" Nom de l'employe : ");
184     viderBuffer();
185     fgets(employe.nom, 22, stdin);
186     // remplace les ' ' par des '_' dans la chaine de caracteres
187     for(j=0; j<strlen(employe.nom); j++) {
188         if(employe.nom[j] == ' ') {
189             employe.nom[j] = '_';
190         }
191         if(employe.nom[j] == '\n') {
192             employe.nom[j] = '\0';
193         }
194     }
195     erreurServiceEmploye://gestion d'erreur
196     system("cls");
197     afficherGestionPersonnelSimple();
198     recupEmployes();
199     printf(" Service de l'employe midi(1), soir(2) : ");
200     testDigit = scanf("%d", &employe.service);
201     if(testDigit==0 || (employe.service != 1 && employe.service != 2)) {
202         viderBuffer();
203         goto erreurServiceEmploye;
204     }
205     system("cls");
206     afficherGestionPersonnelSimple();
207     recupEmployes();
208     printf(" Fonction de l'employe : ");
209     viderBuffer();
210     fgets(employe.fonction, 20, stdin);
211     // remplace les ' ' par des '_' dans la chaine de caracteres
212     for(j=0; j<strlen(employe.fonction); j++) {
213         if(employe.fonction[j] == ' ') {
214             employe.fonction[j] = '_';

```

```
215     }
216     if(employe.fonction[j] == '\0') {
217         employe.fonction[j] = '\0';
218     }
219 }
220 if(compterEmploye()!=0) {
221     fprintf(fdat, "\n");
222 }
223 fprintf(fdat, "%s %d\n", employe.nom, employe.service, employe.fonction);
224 fclose(fdat);
225 }
```



# AuversackHoudart\_compter.c

```

1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <string.h>
4  #include "../Headers/AuversackHoudart_structures.h"
5  int compterMenu() { //Compte le nombre de menus
6      FILE *fdat;
7      fdat = fopen("Data/AuversackHoudart_Menu.dat", "r");
8      int n=0, i, j, fichierVide;
9      Menu *deb, *courant, *suivant;
10     courant=malloc(sizeof(Menu));
11     deb=courant;
12     // Lecture + Construction de ma liste chaînée
13     while(!feof(fdat)) {
14         fichierVide = fscanf(fdat,"%s",&courant->nom);
15         fscanf(fdat,"%s",&courant->prix);
16         fscanf(fdat,"%s",&courant->description);
17         suivant=malloc(sizeof(Menu));
18         courant->suivant=suivant;
19         n++;
20         courant=suivant;
21     }
22     //Placer Null au suivant du dernière élément + libérer l'espace de suivant
23     courant=deb;
24     for(i=1;i<n;i++) {
25         courant=courant->suivant;
26     }
27     courant->suivant=NULL;
28     courant=deb;
29     free(courant);
30     free(suivant);
31     fclose(fdat);
32     if(fichierVide==1) {
33         return 0;
34     }
35     return n;
36 }
37 int compterTable() { //compte le nombre de tables
38     int n=0, i, fichierVide;
39     FILE *fdat;
40     fdat = fopen("Data/AuversackHoudart_Table.dat", "r");
41     Table *deb, *courant, *suivant;
42     courant=malloc(sizeof(Table));
43     deb=courant;
44     // Lecture + Construction de ma liste chaînée
45     while(!feof(fdat)) {
46         fichierVide = fscanf(fdat,"%d",&courant->estReserveMatin);
47         if(courant->estReserveMatin == 1) {
48             fscanf(fdat,"%s",&courant->nomMatin);
49             fscanf(fdat,"%d",&courant->nbPersonneMatin);
50             fscanf(fdat,"%d",&courant->numMenuMatin);
51         }
52         fscanf(fdat,"%d",&courant->estReserveSoir);
53         if(courant->estReserveSoir == 1) {
54             fscanf(fdat,"%s",&courant->nomSoir);
55             fscanf(fdat,"%d",&courant->nbPersonneSoir);
56             fscanf(fdat,"%d",&courant->numMenuSoir);
57         }
58         fscanf(fdat,"%d",&courant->nbPlaceMax);
59         suivant=malloc(sizeof(Table));
60         courant->suivant=suivant;
61         n++;
62         courant=suivant;
63     }
64     //Placer Null au suivant du dernière élément + libérer l'espace de suivant
65     courant=deb;
66     for(i=1;i<n;i++) {
67         courant=courant->suivant;
68     }
69     courant->suivant=NULL;
70     free(courant);
71     free(suivant);
72     fclose(fdat);
73 }

```

```

74     if(fichierVide==0) {
75         return 0;
76     }
77     return n;
78 }
79 int compterTableReservMatin() { //compte le nombre de tables réservées au matin
80     int n=0, i, fichierVide, nMatin=0;
81     FILE *fdat;
82     fdat = fopen("Data/AnversackRoudart_Table.dat", "r");
83     Table *deb, *courant, *suivant;
84     courant=malloc(sizeof(Table));
85     deb=courant;
86     // Lecture + Construction de ma liste chaînée
87     while(!feof(fdat)) {
88         fichierVide = fscanf(fdat,"%d",&courant->estReserveMatin);
89         if(courant->estReserveMatin == 1) {
90             fscanf(fdat,"%s",&courant->nomMatin);
91             fscanf(fdat,"%d",&courant->nbPersonneMatin);
92             fscanf(fdat,"%d",&courant->numMenuMatin);
93             nMatin++;
94         }
95         fscanf(fdat,"%d",&courant->estReserveSoir);
96         if(courant->estReserveSoir == 1) {
97             fscanf(fdat,"%s",&courant->nomSoir);
98             fscanf(fdat,"%d",&courant->nbPersonneSoir);
99             fscanf(fdat,"%d",&courant->numMenuSoir);
100         }
101         fscanf(fdat,"%d",&courant->nbPlaceMax);
102         suivant=malloc(sizeof(Table));
103         courant->suivant=suivant;
104         n++;
105         courant=suivant;
106     }
107     //Placer Null au suivant du dernière élément + libérer l'espace de suivant
108     courant=deb;
109     for(i=0;i<n;i++) {
110         courant=courant->suivant;
111     }
112     courant->suivant=NULL;
113     free(courant);
114     free(suivant);
115     fclose(fdat);
116     if(fichierVide==0) {
117         return 0;
118     }
119     return nMatin;
120 }
121 int compterTableReservSoir() { //compte le nombre de tables réservées au soir
122     int n=0, i, fichierVide, nSoir=0;
123     FILE *fdat;
124     fdat = fopen("Data/AnversackRoudart_Table.dat", "r");
125     Table *deb, *courant, *suivant;
126     courant=malloc(sizeof(Table));
127     deb=courant;
128     // Lecture + Construction de ma liste chaînée
129     while(!feof(fdat)) {
130         fichierVide = fscanf(fdat,"%d",&courant->estReserveMatin);
131         if(courant->estReserveMatin == 1) {
132             fscanf(fdat,"%s",&courant->nomMatin);
133             fscanf(fdat,"%d",&courant->nbPersonneMatin);
134             fscanf(fdat,"%d",&courant->numMenuMatin);
135         }
136         fscanf(fdat,"%d",&courant->estReserveSoir);
137         if(courant->estReserveSoir == 1) {
138             fscanf(fdat,"%s",&courant->nomSoir);
139             fscanf(fdat,"%d",&courant->nbPersonneSoir);
140             fscanf(fdat,"%d",&courant->numMenuSoir);
141             nSoir++;
142         }
143         fscanf(fdat,"%d",&courant->nbPlaceMax);
144         suivant=malloc(sizeof(Table));
145         courant->suivant=suivant;
146         n++;

```

```

147     courant=suivant;
148 }
149 //Placer Null au suivant du dernière élément + libérer l'espace de suivant
150 courant=deb;
151 for(i=1;i<n;i++) {
152     courant=courant->suivant;
153 }
154 courant->suivant=NULL;
155 free(courant);
156 free(suivant);
157 fclose(fdat);
158 if(fichierVide==1) {
159     return 0;
160 }
161 return nSoir;
162 }
163 int compterEmploye() { //compte le nombre d'employé dans la liste
164     int n=0, i, j, fichierVide;
165     FILE *fdat;
166     fdat = fopen("Data/AuversackRoadart_Employes.dat", "r");
167     Employe *deb, *courant, *suivant;
168     courant=malloc(sizeof(Employe));
169     deb=courant;
170     // Lecture + Construction de ma liste chaînée
171     while(!feof(fdat)) {
172         fichierVide = fscanf(fdat,"%s",&courant->nom);
173         fscanf(fdat,"%d",&courant->service);
174         fscanf(fdat,"%s",&courant->fonction);
175         suivant=malloc(sizeof(Employe));
176         courant->suivant=suivant;
177         n++;
178         courant=suivant;
179     }
180     //Placer Null au suivant du dernière élément + libérer l'espace de suivant
181     courant=deb;
182     for(i=1;i<n;i++) {
183         courant=courant->suivant;
184     }
185     courant->suivant=NULL;
186     courant=deb;
187     free(courant);
188     free(suivant);
189     fclose(fdat);
190     if(fichierVide==1) {
191         return 0;
192     }
193     return n;
194 }

```

# AuversackHoudart\_operation.c

```

1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <string.h>
4  #include "../Headers/AuversackHoudart_structures.h"
5  void faireAddition(int tableAddition, int service) { //calcule l'addition d'une
table en fonction de son menu
6      int nbPlaceMaxSelect, reserveSelect, nbPersonneSelect, numMenuSelect;
7      float sommeAddition, prixSelect;
8      char nomPersonneSelect[21], nomMenuSelect[21], descriptionSelect[40];
9      // Traitement table
10     int nTable=0, nMenu=0, i, j;
11     FILE *fdat;
12     fdat = fopen("Data/AuversackHoudart_Table.dat", "r");
13     Table *deb, *courant, *suivant;
14     courant=malloc(sizeof(Table));
15     deb=courant;
16     // Lecture + Construction de ma liste chaînée
17     while(!feof(fdat)) {
18         // Initialisation
19         courant->estReserveMatin = 0;
20         strcpy(courant->nomMatin, "");
21         courant->nbPersonneMatin = 0;
22         courant->numMenuMatin = 0;
23         courant->estReserveSoir = 0;
24         strcpy(courant->nomSoir, "");
25         courant->nbPersonneSoir = 0;
26         courant->numMenuSoir = 0;
27         fscanf(fdat, "%d", &courant->estReserveMatin);
28         if(courant->estReserveMatin == 1) {
29             fscanf(fdat, "%s", &courant->nomMatin);
30             fscanf(fdat, "%d", &courant->nbPersonneMatin);
31             fscanf(fdat, "%d", &courant->numMenuMatin);
32         }
33         fscanf(fdat, "%d", &courant->estReserveSoir);
34         if(courant->estReserveSoir == 1) {
35             fscanf(fdat, "%s", &courant->nomSoir);
36             fscanf(fdat, "%d", &courant->nbPersonneSoir);
37             fscanf(fdat, "%d", &courant->numMenuSoir);
38         }
39         fscanf(fdat, "%d", &courant->nbPlaceMax);
40         suivant=malloc(sizeof(Table));
41         courant->suivant=suivant;
42         nTable++;
43         courant=suivant;
44     }
45     //Placer Null au suivant du dernière élément + libérer l'espace de suivant
46     courant=deb;
47     for(i=1; i<nTable; i++) {
48         courant=courant->suivant;
49     }
50     courant->suivant=NULL;
51     courant=deb;
52     // Affichage
53     for(i=1; i<nTable; i++) {
54         if(i==tableAddition) {
55             if(service==1) {
56                 nbPlaceMaxSelect = courant->nbPlaceMax;
57                 reserveSelect = courant->estReserveMatin;
58                 strcpy(nomPersonneSelect, courant->nomMatin);
59                 numMenuSelect = courant->numMenuMatin;
60                 nbPersonneSelect = courant->nbPersonneMatin;
61             }
62             else if(service==2) {
63                 nbPlaceMaxSelect = courant->nbPlaceMax;
64                 reserveSelect = courant->estReserveSoir;
65                 strcpy(nomPersonneSelect, courant->nomSoir);
66                 numMenuSelect = courant->numMenuSoir;
67                 nbPersonneSelect = courant->nbPersonneSoir;
68             }
69         }
70         courant=courant->suivant;
71     }
72     free(courant);

```



```

73 free(suivant);
74 fclose(fdat);
75 // Traitement Menu
76 FILE *fdat2;
77 fdat2 = fopen("Data/AnversackHoudart_Menu.dat", "r");
78 Menu *deb2, *courant2, *suivant2;
79 courant2=malloc(sizeof(Menu));
80 deb2=courant2;
81 // Lecture + Construction de ma liste chaînée
82 while(!feof(fdat)) {
83     fscanf(fdat2,"%s",&courant2->nom);
84     fscanf(fdat2,"%i",&courant2->prix);
85     fscanf(fdat2,"%s",&courant2->description);
86     suivant2=malloc(sizeof(Menu));
87     courant2->suivant=suivant2;
88     nMenu++;
89     courant2=suivant2;
90 }
91 //Placer Null au suivant du dernière élément + libérer l'espace de suivant
92 courant2=deb2;
93 for(i=0;i<nMenu;i++) {
94     courant2=courant2->suivant;
95 }
96 courant2->suivant=NULL;
97 courant2=deb2;
98 // Affichage
99 for(i=0;i<nMenu;i++) {
100     if(i==numMenuSelect) {
101         for(j=0;j<strlen(courant2->description);j++) {
102             if(courant2->description[j] == ' ') {
103                 courant2->description[j] = ' ';
104             }
105         }
106         strcpy(numMenuSelect, courant2->nom);
107         prixSelect = courant2->prix;
108         strcpy(descriptionSelect, courant2->description);
109     }
110     courant2=courant2->suivant;
111 }
112 if(tableAddition>nTable || tableAddition<=0 || reserveSelect==0) {
113     goto erreurTailleTable;
114 }
115 //affichage de l'addition d'une table
116 sommeAddition = prixSelect * nbPersonneSelect;
117 system("cls");
118 printf("\n");
119 printf(".....");
120 printf("\n");
121 printf(".....\n");
122 printf(".....\n");
123 printf(".....\n");
124 printf(".....\n");
125 printf(".....\n");
126 printf(".....\n");
127 printf("\n");
128 printf(".....");
129 printf("\n");
130 printf("\n");
131 if(service==1) {
132     printf("Table\n");
133     printf("
134     --\n");
135     printf(" | Num table | nb place max | Reserve | Num | nb
136     Pers. | Num menu |\n");
137     printf("
138     --\n");
139     printf(" | %d | %2d | %-3s | %-12s | %d
140     | %d |\n",tableAddition, nbPlaceMaxSelect, (reserveSelect ?
141     "Oui":"Non"), nomPersonneSelect, nbPersonneSelect, numMenuSelect);
142     printf("

```

```

138     --\n\n");
139     printf("      (Menu)\n");
140     printf("      \n");
141     printf("      | N |      Nom |      Prix (Euro) |
Description
      | \n");
142     printf("      \n");
143     printf("      | %d | %-20s | %5.2f | %-54s|\n", numMenuSelect,
nomMenuSelect, prixSelect, descriptionSelect);
144     printf("      \n");
145 }
146 else if(service==2) {
147     printf("      (Table)\n");
148     printf("      \n");
149     printf("      | Num table | nb place max | Reserve |      Nom | nb
Pers. | Num menu |\n");
150     printf("      \n");
151     printf("      | %d | %2d | %-3d | %-12s | %d
| %d | \n", tableAddition, nbPlaceMaxSelect, {reserveSelect ?
"Oui": "Non"}, nomPersonneSelect, nbPersonneSelect, numMenuSelect);
152     printf("      \n");
153     printf("      (Menu)\n");
154     printf("      \n");
155     printf("      | N |      Nom |      Prix (Euro) |
Description
      | \n");
156     printf("      \n");
157     printf("      | %d | %-20s | %5.2f | %-54s|\n", numMenuSelect,
nomMenuSelect, prixSelect, descriptionSelect);
158     printf("      \n");
159 }
160 printf("\n      (Addition)\n");
161 printf("      \n");
162 printf("      | L'addition de la table %d est de %5.2f Euro. |\n", tableAddition,
sommeAddition);
163 printf("      \n");
164 erreurTailleTable: //gestion d'erreur
165 free(courant2);
166 free(suivant2);
167 fclose(fdat2);
168 }
169 void changerCommande() { //permet de changer le menu commander par une table en
utilisant remplacerMenuTable
170     int service, numTable, numMenuChoisi;
171     erreurIndexService: //gestion d'erreur
172     system("cls");
173     afficherChangerCommande();
174     recupTableReserveMidi();
175     recupTableReserveSoir();
176     printf("      Service (Midi=1/Soir=2) : ");
177     scanf("%d", &service);
178     if((service != 1 && service != 2) || {service == 1 &&
compterTableReservMatin()==0} || {service == 2 && compterTableReservSoir()==0}) {
179         viderBuffer();
180         goto erreurIndexService;
181     }

```

```

182 erreurTableNonReserve: //gestion d'erreur
183 system("cls");
184 afficherChangerCommande();
185 if(service==1) {
186     recupTableReserveMidi();
187     printf("    Numero de votre table : ");
188     scanf("%d", &numTable);
189     if(estReserve(1, numTable)==0) {
190         viderBuffer();
191         goto erreurTableNonReserve;
192     }
193     erreurIndexNumMenuMidi: //gestion d'erreur
194     system("cls");
195     afficherChangerCommande();
196     selectTable(numTable, 1);
197     printf("    Menu");
198     recupMenu();
199     printf("    Quel menu voulez vous ? : ");
200     scanf("%d", &numMenuChoisi);
201     if(numMenuChoisi<=0 || numMenuChoisi > compterMenu()) {
202         viderBuffer();
203         goto erreurIndexNumMenuMidi;
204     }
205     printf("\n");
206     replaceMenuTable(numMenuChoisi, numTable, service);
207     printf("    Vous avez choisi le menu : %s\n\n", nomMenuChoisi[numMenuChoisi]);
208 }
209 else if(service==2) {
210     recupTableReserveSoir();
211     printf("    Numero de votre table : ");
212     scanf("%d", &numTable);
213     if(estReserve(2, numTable)==0) {
214         viderBuffer();
215         goto erreurTableNonReserve;
216     }
217     erreurIndexNumMenuSoir: //gestion d'erreur
218     system("cls");
219     afficherChangerCommande();
220     selectTable(numTable, 2);
221     printf("    Menu");
222     recupMenu();
223     printf("    Quel menu voulez vous ? : ");
224     scanf("%d", &numMenuChoisi);
225     if(numMenuChoisi<=0 || numMenuChoisi > compterMenu()) {
226         viderBuffer();
227         goto erreurIndexNumMenuSoir;
228     }
229     printf("\n");
230     replaceMenuTable(numMenuChoisi, numTable, service);
231     printf("    Vous avez choisi le menu : %s\n\n", nomMenuChoisi[numMenuChoisi]);
232 }
233 }
234 void faireReservation() { //Permet de reserver une table libre
235     int service, nbPersonne, numMenu, reserveOk = 0;
236     char non[20];
237     erreurIndexService: //gestion d'erreur
238     system("cls");
239     afficherReservation();
240     recupTableLibreMidi();
241     recupTableLibreSoir();
242     printf("\n");
243     printf("    Service (Midi=1/Soir=2) : ");
244     scanf("%d", &service);
245     if(service != 1 && service != 2) {
246         viderBuffer();
247         goto erreurIndexService;
248     }
249     system("cls");
250     afficherReservation();
251     if(service==1) {
252         recupTableLibreMidi();
253         printf("    Nom de la reservation : ");
254         scanf("%s", &non);

```

```

255     erreurIndexnbPersonne: //gestion d'erreur
256     system("cls");
257     afficherReservation();
258     recupTableLibreMidi();
259     printf("  Nombre de personnes : ");
260     scanf("%d", &nbPersonne);
261     if(nbPersonne<=0) {
262         viderBuffer();
263         goto erreurIndexnbPersonne;
264     }
265     erreurIndexNumMenuMidi: //gestion d'erreur
266     system("cls");
267     afficherReservation();
268     recupTableLibreMidi();
269     printf("  Menu:");
270     recupMenu();
271     printf("  Numero du menu : ");
272     scanf("%d", &numMenu);
273     if(numMenu<=0 || numMenu > compterMenu()) {
274         viderBuffer();
275         goto erreurIndexNumMenuMidi;
276     }
277     reserveOk = remplaceNonReserveTable(rechercheTableLibre[1], nbPersonne, nom,
278     nbPersonne, numMenu, 1);
279     if(reserveOk==0) {
280         printf("\n  Pas de table disponible !\n\n");
281     }
282     else {
283         printf("\n  Reservation reussie !\n\n");
284     }
285 }
286 else if(service==0) {
287     recupTableLibreSoir();
288     printf("  Nom de la reservation : ");
289     scanf("%s", &nom);
290     system("cls");
291     afficherReservation();
292     recupTableLibreSoir();
293     printf("  Nombre de personnes : ");
294     scanf("%d", &nbPersonne);
295     erreurIndexNumMenuSoir: //gestion d'erreur
296     system("cls");
297     afficherReservation();
298     recupTableLibreSoir();
299     printf("  Menu:");
300     recupMenu();
301     printf("  Numero du menu : ");
302     scanf("%d", &numMenu);
303     if(numMenu<=0 || numMenu > compterMenu()) {
304         viderBuffer();
305         goto erreurIndexNumMenuSoir;
306     }
307     reserveOk = remplaceNonReserveTable(rechercheTableLibre[2], nbPersonne, nom,
308     nbPersonne, numMenu, 2);
309     if(reserveOk==0) {
310         printf("\n  Pas de table disponible !\n\n");
311     }
312     else {
313         printf("\n  Reservation reussie !\n\n");
314     }
315 }
316 }
317 void modifierEmploye() { //permet de modifier un parametre d'un employe
318     int numEmploye, choixModif, serviceModif, testDigit;
319     char fonctionModif[21], nomModif[21];
320     erreurSelectEmploye: //gestion d'erreur
321     system("cls");
322     afficherGestionPersonnelSimple();
323     recupEmployes();
324     printf("  Selectionner un employe : ");
325     testDigit = scanf("%d", &numEmploye);
326     if(testDigit==0 || numEmploye<=0 || numEmploye>compterEmploye()) {
327         viderBuffer();

```



```

326         goto erreurSelectEmploye;
327     }
328     erreurChoixModif://gestion d'erreur
329     system("cls");
330     afficherGestionPersonnelSimple();
331     selectEmploye(numEmploye);
332     printf(" 1. Modifier le nom\n");
333     printf(" 2. Modifier le service\n");
334     printf(" 3. Modifier la fonction\n\n");
335     printf(" Choix : ");
336     testDigit = scanf("%d", &choixModif);
337     if(testDigit==0 || choixModif<=0 || choixModif>3) {
338         viderBuffer();
339         goto erreurChoixModif;
340     }
341     if(choixModif==1) {
342         system("cls");
343         afficherGestionPersonnelSimple();
344         selectEmploye(numEmploye);
345         printf(" Nom de l'employe : ");
346         scanf("%s", &nonModif);
347         //Modifier le nom de l'employe
348         modifierNomEmploye(numEmploye, nonModif);
349         printf("\n\n Modification réussie !");
350     }
351     else if(choixModif==2) {
352         erreurServiceModif://gestion d'erreur
353         system("cls");
354         afficherGestionPersonnelSimple();
355         selectEmploye(numEmploye);
356         printf(" Service du midi(1), service du soir(2) : ");
357         testDigit = scanf("%d", &serviceModif);
358         if(testDigit==0 || serviceModif<=0 || serviceModif>2) {
359             viderBuffer();
360             goto erreurServiceModif;
361         }
362         //Modifier Service
363         modifierService(numEmploye, serviceModif);
364         printf("\n\n Modification réussie !");
365     }
366     else if(choixModif==3) {
367         system("cls");
368         afficherGestionPersonnelSimple();
369         selectEmploye(numEmploye);
370         printf(" Nom de la fonction : ");
371         scanf("%s", &fonctionModif);
372         //Modifier fonction
373         modifierFonction(numEmploye, fonctionModif);
374         printf("\n\n Modification réussie !");
375     }
376     else {}

```

# AuversackHoudart\_ouils.c

```

1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <string.h>
4  #include "../Headers/AuversackHoudart_structures.h"
5  int estReserve(int service, int numTable) { //Retourne un entier qui permet de
    savoir si la table est reservee ou non
6      int n=0, i;
7      FILE *fdat, *fdatTmp;
8      fdat = fopen("Data/AuversackHoudart_Table.dat", "r");
9      Table *deb, *courant, *suivant;
10     courant=malloc(sizeof(Table));
11     deb=courant;
12     // Lecture + Construction de ma liste chainee
13     while(!feof(fdat)) {
14         courant->estReserveMatin = 0;
15         strcpy(courant->nomMatin, "");
16         courant->nbPersonneMatin = 0;
17         courant->numMenuMatin = 0;
18         courant->estReserveSoir = 0;
19         strcpy(courant->nomSoir, "");
20         courant->nbPersonneSoir = 0;
21         courant->numMenuSoir = 0;
22         fscanf(fdat, "%d", &courant->estReserveMatin);
23         if(courant->estReserveMatin == 1) {
24             fscanf(fdat, "%s", &courant->nomMatin);
25             fscanf(fdat, "%d", &courant->nbPersonneMatin);
26             fscanf(fdat, "%d", &courant->numMenuMatin);
27         }
28         fscanf(fdat, "%d", &courant->estReserveSoir);
29         if(courant->estReserveSoir == 1) {
30             fscanf(fdat, "%s", &courant->nomSoir);
31             fscanf(fdat, "%d", &courant->nbPersonneSoir);
32             fscanf(fdat, "%d", &courant->numMenuSoir);
33         }
34         fscanf(fdat, "%d", &courant->nbPlaceMax);
35         suivant=malloc(sizeof(Table));
36         courant->suivant=suivant;
37         n++;
38         courant=suivant;
39     }
40     //Placer Null au suivant du dernière élément + libérer l'espace de suivant
41     courant=deb;
42     for(i=1; i<=n; i++) {
43         courant->suivant=NULL;
44         courant=deb;
45     }
46     // Recherche
47     for(i=1; i<=n; i++) {
48         if(service==1 && i==numTable) {
49             if(courant->estReserveMatin == 1) {
50                 fclose(fdat);
51                 return 1;
52             }
53         }
54         else if(service==2 && i==numTable) {
55             if(courant->estReserveSoir == 1) {
56                 fclose(fdat);
57                 return 1;
58             }
59         }
60         courant=courant->suivant;
61     }
62     fclose(fdat);
63     return 0;
64 }
65
66 const char* nonMenuChoisi(int numMenu) { //Retourne le nom d'un menu precis
67     static char nonMenuChoisi[21];
68     int n=0, i, j;
69     FILE *fdat;
70     fdat = fopen("Data/AuversackHoudart_Menu.dat", "r");
71     Menu *deb, *courant, *suivant;
72

```

```

73     courant=malloc(sizeof(Menu));
74     deb=courant;
75     // Lecture + Construction de ma liste chaînée
76     while(!feof(fdat)) {
77         fscanf(fdat,"%s",&courant->nom);
78         fscanf(fdat,"%i",&courant->prix);
79         fscanf(fdat,"%s",&courant->description);
80         suivant=malloc(sizeof(Menu));
81         courant->suivant=suivant;
82         n++;
83         courant=suivant;
84     }
85     //Placer Null au suivant du dernière élément + libérer l'espace de suivant
86     courant=deb;
87     for(i=1;i<n;i++) {
88         courant=courant->suivant;
89     }
90     courant->suivant=NULL;
91     courant=deb;
92     for(i=1;i<=n;i++) {
93         for(j=0; j<strlen(courant->description); j++) {
94             if(courant->description[j] == ' ') {
95                 courant->description[j] = '_';
96             }
97         }
98         if(i==numMenu) {
99             strcpy(nonMenuChoisi, courant->nom);
100         }
101         courant=courant->suivant;
102     }
103     free(courant);
104     free(suivant);
105     fclose(fdat);
106     return nonMenuChoisi;
107 }
108 void remplaceMenuTable(int numMenu, int numTable, int service) { //change le menu
d'une table precise
109     int n=0, i;
110     FILE *fdat, *fdatTnp;
111     fdat = fopen("Data/AuversackHoodart_Table.dat", "r");
112     fdatTnp = fopen("Data/AuversackHoodart_Table.tnp", "w");
113     Table *deb, *courant, *suivant;
114     courant=malloc(sizeof(Table));
115     deb=courant;
116     // Lecture + Construction de ma liste chaînée
117     while(!feof(fdat)) {
118         courant->estReserveMatin = 0;
119         strcpy(courant->nomMatin, "");
120         courant->nbPersonneMatin = 0;
121         courant->numMenuMatin = 0;
122         courant->estReserveSoir = 0;
123         strcpy(courant->nomSoir, "");
124         courant->nbPersonneSoir = 0;
125         courant->numMenuSoir = 0;
126         fscanf(fdat,"%d",&courant->estReserveMatin);
127         if(courant->estReserveMatin == 1) {
128             fscanf(fdat,"%s",&courant->nomMatin);
129             fscanf(fdat,"%d",&courant->nbPersonneMatin);
130             fscanf(fdat,"%d",&courant->numMenuMatin);
131         }
132         fscanf(fdat,"%d",&courant->estReserveSoir);
133         if(courant->estReserveSoir == 1) {
134             fscanf(fdat,"%s",&courant->nomSoir);
135             fscanf(fdat,"%d",&courant->nbPersonneSoir);
136             fscanf(fdat,"%d",&courant->numMenuSoir);
137         }
138         fscanf(fdat,"%d",&courant->nbPlaceMax);
139         suivant=malloc(sizeof(Table));
140         courant->suivant=suivant;
141         n++;
142         courant=suivant;
143     }
144 }

```

```

145 //Placer Null au suivant du dernière élément + libérer l'espace de suivant
146 courant=deb;
147 for(i=1;i<n;i++) {
148     courant=courant->suitant;
149 }
150 courant->suitant=NULL;
151 courant=deb;
152 // Ecriture
153 for(i=1;i<n;i++) {
154     fprintf(fdatTnp, "%d ", courant->estReserveMatin);
155     if(courant->estReserveMatin==1) {
156         if(numTable==1 && service==1) {
157             courant->numMenuMatin = numMenu;
158         }
159         fprintf(fdatTnp, "%s %d %d ", courant->nomMatin,
            courant->nbPersonneMatin, courant->numMenuMatin);
160     }
161     fprintf(fdatTnp, "%d", courant->estReserveSoir);
162     if(courant->estReserveSoir==1) {
163         if(numTable==1 && service==1) {
164             courant->numMenuSoir = numMenu;
165         }
166         fprintf(fdatTnp, " %s %d %d", courant->nomSoir, courant->nbPersonneSoir,
            courant->numMenuSoir);
167     }
168     fprintf(fdatTnp, " %d", courant->nbPlaceMax);
169     if(i!=n) {
170         fprintf(fdatTnp, "\n");
171     }
172     courant=courant->suitant;
173 }
174 fclose(fdat);
175 fclose(fdatTnp);
176 remove("Data/AuversackHoudart_Table.dat");
177 rename("Data/AuversackHoudart_Table.tmp", "Data/AuversackHoudart_Table.dat");
178 }
179 void selectTable(int numTable, int service) { //affiche une table precise
180     int nbPlaceMaxSelect, reserveSelect, nbPersonneSelect, numMenuSelect, nTable=0,
        i, j;
181     char nomPersonneSelect[51];
182     // Traitement table
183     FILE *fdat;
184     fdat = fopen("Data/AuversackHoudart_Table.dat", "r");
185     Table *deb, *courant, *suitant;
186     courant=malloc(sizeof(Table));
187     deb=courant;
188     // Lecture + Construction de ma liste chaînée
189     while(!feof(fdat)) {
190         // Initialisation
191         courant->estReserveMatin = 0;
192         strcpy(courant->nomMatin, "");
193         courant->nbPersonneMatin = 0;
194         courant->numMenuMatin = 0;
195         courant->estReserveSoir = 0;
196         strcpy(courant->nomSoir, "");
197         courant->nbPersonneSoir = 0;
198         courant->numMenuSoir = 0;
199         fscanf(fdat, "%d", &courant->estReserveMatin);
200         if(courant->estReserveMatin == 1) {
201             fscanf(fdat, "%s", &courant->nomMatin);
202             fscanf(fdat, "%d", &courant->nbPersonneMatin);
203             fscanf(fdat, "%d", &courant->numMenuMatin);
204         }
205         fscanf(fdat, "%d", &courant->estReserveSoir);
206         if(courant->estReserveSoir == 1) {
207             fscanf(fdat, "%s", &courant->nomSoir);
208             fscanf(fdat, "%d", &courant->nbPersonneSoir);
209             fscanf(fdat, "%d", &courant->numMenuSoir);
210         }
211         fscanf(fdat, "%d", &courant->nbPlaceMax);
212         suitant=malloc(sizeof(Table));
213         courant->suitant=suitant;
214         nTable++;

```



```

215     courant=suivant;
216 }
217 //Placer Null au suivant du dernière élément + libérer l'espace de suivant
218 courant=deb;
219 for(i=0;i<nTable;i++) {
220     courant=courant->suivant;
221 }
222 courant->suivant=NULL;
223 courant=deb;
224 // Affichage
225 for(i=0;i<nTable;i++) {
226     if(i==numTable) {
227         if(service==1) {
228             nbPlaceMaxSelect = courant->nbPlaceMax;
229             reserveSelect = courant->estReserveMatin;
230             strcpy(nonPersonneSelect, courant->nomMatin);
231             numMenuSelect = courant->numMenuMatin;
232             nbPersonneSelect = courant->nbPersonneMatin;
233         }
234         else if(service==2) {
235             nbPlaceMaxSelect = courant->nbPlaceMax;
236             reserveSelect = courant->estReserveSoir;
237             strcpy(nonPersonneSelect, courant->nomSoir);
238             numMenuSelect = courant->numMenuSoir;
239             nbPersonneSelect = courant->nbPersonneSoir;
240         }
241     }
242     courant=courant->suivant;
243 }
244 free(courant);
245 free(suivant);
246 fclose(fdat);
247 // Affichage
248 if(service==1) {
249     printf("\n");
250     printf("    Table\n");
251     printf("-----\n");
252     printf("    | Num table | nb place max: | Reserve | Nom | nb Pers. | Num menu |\n");
253     printf("-----\n");
254     printf("    | %d | %2d | %s | %s | %d | %d |\n", numTable, nbPlaceMaxSelect, (reserveSelect ? "Oui": "Non"), nonPersonneSelect, nbPersonneSelect, numMenuSelect);
255     printf("-----\n");
256 }
257 else if(service==2) {
258     printf("\n");
259     printf("    Table\n");
260     printf("-----\n");
261     printf("    | Num table | nb place max: | Reserve | Nom | nb Pers. | Num menu |\n");
262     printf("-----\n");
263     printf("    | %d | %2d | %s | %s | %d | %d |\n", numTable, nbPlaceMaxSelect, (reserveSelect ? "Oui": "Non"), nonPersonneSelect, nbPersonneSelect, numMenuSelect);
264     printf("-----\n");
265 }
266 }
267 void selectEmploye(int numEmploye) { //affiche un employe précis
268     int serviceSelect, nEmploye=0, i, j;
269     char nomEmployeSelect[20], nomFonctionSelect[15];

```

```

270 // Traitement table
271 FILE *fdat;
272 fdat = fopen("Data/AuversackHoudart_Employes.dat", "r");
273 Employe *deb, *courant, *suivant;
274 courant = malloc(sizeof(Employe));
275 deb = courant;
276 // Lecture + Construction de ma liste chaînée
277 while(!feof(fdat)) {
278     // Initialisation
279     strcpy(courant->nom, "");
280     courant->service = 0;
281     strcpy(courant->fonction, "");
282     fscanf(fdat, "%s", &courant->nom);
283     fscanf(fdat, "%d", &courant->service);
284     fscanf(fdat, "%s", &courant->fonction);
285     suivant = malloc(sizeof(Employe));
286     courant->suivant = suivant;
287     nEmploye++;
288     courant = suivant;
289 }
290 //Placer Null au suivant du dernière élément + libérer l'espace de suivant
291 courant = deb;
292 for(i=1; i<=nEmploye; i++) {
293     courant = courant->suivant;
294 }
295 courant->suivant = NULL;
296 courant = deb;
297 // Affichage
298 for(i=1; i<=nEmploye; i++) {
299     if(i==nEmploye) {
300         strcpy(nonEmployeSelect, courant->nom);
301         serviceSelect = courant->service;
302         strcpy(nonFonctionSelect, courant->fonction);
303     }
304     courant = courant->suivant;
305 }
306 free(courant);
307 free(suivant);
308 fclose(fdat);
309 // Affichage
310 printf("\n");
311 printf("    Service - Midi : 12h00-15h00 - Soir : 18h00-23h00");
312 printf("\n");
313 printf("\n");
314 printf("    Employe:\n");
315 printf("-----\n");
316 printf(" | N |      Nom      | Service | Fonction | \n");
317 printf("-----\n");
318 printf(" | id | %-10s | %-6s | %-10s | \n", numEmploye,
nonEmployeSelect, (serviceSelect==1)?"Midi":(serviceSelect==2)?"Soir":"Aucun",
nonFonctionSelect);
319 printf("-----\n");
320 printf("\n");
321 }
322 int rechercheTableLibre(int service, int nbPersonne) { //recherche toutes les tables
libres dans la liste des tables
323 int n=0, i;
324 FILE *fdat, *fdatTmp;
325 fdat = fopen("Data/AuversackHoudart_Table.dat", "r");
326 Table *deb, *courant, *suivant;
327 courant = malloc(sizeof(Table));
328 deb = courant;
329 // Lecture + Construction de ma liste chaînée
330 while(!feof(fdat)) {
331     courant->estReserveMatin = 0;
332     strcpy(courant->nomMatin, "");
333     courant->nbPersonneMatin = 0;
334     courant->numMenuMatin = 0;
335     courant->estReserveSoir = 0;
336     strcpy(courant->nomSoir, "");
337     courant->nbPersonneSoir = 0;
338     courant->numMenuSoir = 0;
339     fscanf(fdat, "%d", &courant->estReserveMatin);

```

```

340     if(courant->estReserveMatin == 1) {
341         fscanf(fdat, "%s", &courant->nomMatin);
342         fscanf(fdat, "%d", &courant->nbPersonneMatin);
343         fscanf(fdat, "%d", &courant->numMenuMatin);
344     }
345     fscanf(fdat, "%d", &courant->estReserveSoir);
346     if(courant->estReserveSoir == 1) {
347         fscanf(fdat, "%s", &courant->nomSoir);
348         fscanf(fdat, "%d", &courant->nbPersonneSoir);
349         fscanf(fdat, "%d", &courant->numMenuSoir);
350     }
351     fscanf(fdat, "%d", &courant->nbPlaceMax);
352     suivant=malloc(sizeof(Table));
353     courant->suivant=suivant;
354     n++;
355     courant=suivant;
356 }
357 //Placer Null au suivant du dernière élément + libérer l'espace de suivant
358 courant=deb;
359 for(i=1; i<n; i++) {
360     courant=courant->suivant;
361 }
362 courant->suivant=NULL;
363 courant=deb;
364 // Recherche
365 for(i=1; i<=n; i++) {
366     if(service==1 && nbPersonne<=courant->nbPlaceMax) {
367         if(courant->estReserveMatin == 0) {
368             fclose(fdat);
369             return i;
370         }
371     }
372     else if(service==3 && nbPersonne<=courant->nbPlaceMax) {
373         if(courant->estReserveSoir == 0) {
374             fclose(fdat);
375             return i;
376         }
377     }
378     courant=courant->suivant;
379 }
380 fclose(fdat);
381 return 0;
382 }
383 int remplaceNonReserveTable(int numTable, char nom[], int nbPersonne, int numMenu,
int service) { //Effectue la réservation sur base des données entrée par
l'utilisateur qui lui sont envoyées
384     int n=0, i, reservOk=0;
385     FILE *fdat, *fdatTmp;
386     fdat = fopen("Data/AuversackHoudart_Table.dat", "r");
387     fdatTmp = fopen("Data/AuversackHoudart_Table.tmp", "w");
388     Table *deb, *courant, *suivant;
389     courant=malloc(sizeof(Table));
390     deb=courant;
391     // Lecture + Construction de ma liste chaînée
392     while(!feof(fdat)) {
393         courant->estReserveMatin = 0;
394         strcpy(courant->nomMatin, "");
395         courant->nbPersonneMatin = 0;
396         courant->numMenuMatin = 0;
397         courant->estReserveSoir = 0;
398         strcpy(courant->nomSoir, "");
399         courant->nbPersonneSoir = 0;
400         courant->numMenuSoir = 0;
401         fscanf(fdat, "%d", &courant->estReserveMatin);
402         if(courant->estReserveMatin == 1) {
403             fscanf(fdat, "%s", &courant->nomMatin);
404             fscanf(fdat, "%d", &courant->nbPersonneMatin);
405             fscanf(fdat, "%d", &courant->numMenuMatin);
406         }
407         fscanf(fdat, "%d", &courant->estReserveSoir);
408         if(courant->estReserveSoir == 1) {
409             fscanf(fdat, "%s", &courant->nomSoir);
410             fscanf(fdat, "%d", &courant->nbPersonneSoir);

```

```

411         fscanf(fdat, "%d", &courant->numMenuSoir);
412     }
413     fscanf(fdat, "%d", &courant->nbPlaceMax);
414     suivant=malloc(sizeof(Table));
415     courant->suivant=suivant;
416     n++;
417     courant=suivant;
418 }
419 //Placer Null au suivant du dernière élément + libérer l'espace de suivant
420 courant=deb;
421 for(i=1;i<n;i++) {
422     courant=courant->suivant;
423 }
424 courant->suivant=NULL;
425 courant=deb;
426 // Ecriture
427 for(i=1;i<=n;i++) {
428     if(courant->estReserveMatin==1) {
429         fprintf(fdatTmp, "%d ", courant->estReserveMatin);
430         fprintf(fdatTmp, "%s %d %d ", courant->nomMatin,
            courant->nbPersonneMatin, courant->numMenuMatin);
431     }
432     if(courant->estReserveMatin==0) {
433         if(numTable==i && service==1) {
434             reservOk = 1;
435             courant->estReserveMatin = 1;
436             courant->nbPersonneMatin = nbPersonne;
437             strcpy(courant->nomMatin, nom);
438             courant->numMenuMatin = numMenu;
439             fprintf(fdatTmp, "%d ", courant->estReserveMatin);
440             fprintf(fdatTmp, "%s %d %d ", courant->nomMatin,
                courant->nbPersonneMatin, courant->numMenuMatin);
441         }
442         else {
443             fprintf(fdatTmp, "%d ", courant->estReserveMatin);
444         }
445     }
446     if(courant->estReserveSoir==1) {
447         fprintf(fdatTmp, "%d ", courant->estReserveSoir);
448         fprintf(fdatTmp, "%s %d %d ", courant->nomSoir, courant->nbPersonneSoir,
            courant->numMenuSoir);
449     }
450     if(courant->estReserveSoir==0) {
451         if(numTable==i && service==2) {
452             reservOk = 1;
453             courant->estReserveSoir = 1;
454             courant->nbPersonneSoir = nbPersonne;
455             strcpy(courant->nomSoir, nom);
456             courant->numMenuSoir = numMenu;
457             fprintf(fdatTmp, "%d ", courant->estReserveSoir);
458             fprintf(fdatTmp, "%s %d %d ", courant->nomSoir,
                courant->nbPersonneSoir, courant->numMenuSoir);
459         }
460         else {
461             fprintf(fdatTmp, "%d", courant->estReserveSoir);
462         }
463     }
464     fprintf(fdatTmp, " %d", courant->nbPlaceMax);
465     if(i!=n) {
466         fprintf(fdatTmp, "\n");
467     }
468     courant=courant->suivant;
469 }
470 fclose(fdat);
471 fclose(fdatTmp);
472 remove("Data/AuversackHoudart_Table.dat");
473 rename("Data/AuversackHoudart_Table.tmp", "Data/AuversackHoudart_Table.dat");
474 return reservOk;
475 }
476 int modifTableSuppRes(int service, int numTable) { //supprime une reservation sur
une table precise
477     int n=0, i, modifOk=0;
478     FILE *fdat, *fdatTmp;

```



```

479 fdat = fopen("Data/AuversackHoudart_Table.dat", "r");
480 fdatTmp = fopen("Data/AuversackHoudart_Table.tmp", "w");
481 Table *deb, *courant, *suivant;
482 courant = malloc(sizeof(Table));
483 deb = courant;
484 // Lecture + Construction de ma liste chaînée
485 while(!feof(fdat)) {
486     courant->estReserveMatin = 0;
487     strcpy(courant->nomMatin, "");
488     courant->nbPersonneMatin = 0;
489     courant->numMenuMatin = 0;
490     courant->estReserveSoir = 0;
491     strcpy(courant->nomSoir, "");
492     courant->nbPersonneSoir = 0;
493     courant->numMenuSoir = 0;
494     fscanf(fdat, "%d", &courant->estReserveMatin);
495     if(courant->estReserveMatin == 1) {
496         fscanf(fdat, "%s", &courant->nomMatin);
497         fscanf(fdat, "%d", &courant->nbPersonneMatin);
498         fscanf(fdat, "%d", &courant->numMenuMatin);
499     }
500     fscanf(fdat, "%d", &courant->estReserveSoir);
501     if(courant->estReserveSoir == 1) {
502         fscanf(fdat, "%s", &courant->nomSoir);
503         fscanf(fdat, "%d", &courant->nbPersonneSoir);
504         fscanf(fdat, "%d", &courant->numMenuSoir);
505     }
506     fscanf(fdat, "%d", &courant->nbPlaceMax);
507     suivant = malloc(sizeof(Table));
508     courant->suivant = suivant;
509     n++;
510     courant = suivant;
511 }
512 //Placer Null au suivant du dernière élément + libérer l'espace de suivant
513 courant = deb;
514 for(i=1; i<=n; i++) {
515     courant = courant->suivant;
516 }
517 courant->suivant = NULL;
518 courant = deb;
519 // Ecriture
520 for(i=1; i<=n; i++) {
521     if(courant->estReserveMatin==1) {
522         if(i==numTable && service == 1) {
523             modifOk=1;
524             courant->estReserveMatin=0;
525             courant->nbPersonneMatin=0;
526             strcpy(courant->nomMatin, "");
527             courant->numMenuMatin = 0;
528         }
529     }
530     fprintf(fdatTmp, "%d ", courant->estReserveMatin);
531     if(courant->estReserveMatin==1) {
532         fprintf(fdatTmp, "%s %d %d ", courant->nomMatin,
533             courant->nbPersonneMatin, courant->numMenuMatin);
534     }
535     if(courant->estReserveSoir==1) {
536         if(i==numTable && service == 2) {
537             courant->estReserveSoir=0;
538             courant->nbPersonneSoir=0;
539             strcpy(courant->nomSoir, "");
540             courant->numMenuSoir = 0;
541             modifOk=1;
542         }
543     }
544     fprintf(fdatTmp, "%d ", courant->estReserveSoir);
545     if(courant->estReserveSoir==1) {
546         fprintf(fdatTmp, "%s %d %d ", courant->nomSoir, courant->nbPersonneSoir,
547             courant->numMenuSoir);
548     }
549     fprintf(fdatTmp, " %d", courant->nbPlaceMax);
550     if(i!=n) {
551         fprintf(fdatTmp, "\n");
552     }
553 }

```

```

550     }
551     courant=courant->suitant;
552 }
553 fclose(fdat);
554 fclose(fdatTmp);
555 remove("Data/AuversackHoudart_Table.dat");
556 rename("Data/AuversackHoudart_Table.tmp", "Data/AuversackHoudart_Table.dat");
557 return modifOk;
558 }
559 void viderBuffer() { //vide le buffer, aide principalement à la gestion d'erreur
    lorsque des lettres sont entrées à la place des chiffres
560     int c;
561     while ((c = getchar()) != '\n' && c != EOF);
562 }
563 void modifierService(int numEmploye, int newService) { //modifie le service d'un
    employé
564     int n=0, i;
565     FILE *fdat, *fdatTmp;
566     fdat = fopen("Data/AuversackHoudart_Employes.dat", "r");
567     fdatTmp = fopen("Data/AuversackHoudart_Employes.tmp", "w");
568     Employe *deb, *courant, *suitant;
569     courant=malloc(sizeof(Employe));
570     deb=courant;
571     // Lecture + Construction de ma liste chaînée
572     while(!feof(fdat)) {
573         fscanf(fdat, "%s", &courant->nom);
574         fscanf(fdat, "%d", &courant->service);
575         fscanf(fdat, "%s", &courant->fonction);
576         suitant=malloc(sizeof(Employe));
577         courant->suitant=suitant;
578         n++;
579         courant=suitant;
580     }
581     //Placer Null au suivant du dernière élément + libérer l'espace de suivant
582     courant=deb;
583     for(i=0; i<n; i++) {
584         courant=courant->suitant;
585     }
586     courant->suitant=NULL;
587     courant=deb;
588     // Ecriture
589     for(i=0; i<n; i++) {
590         if(i==numEmploye) {
591             courant->service = newService;
592         }
593         fprintf(fdatTmp, "%s %d %s", courant->nom, courant->service,
            courant->fonction);
594         if(i!=n) {
595             fprintf(fdatTmp, "\n");
596         }
597         courant=courant->suitant;
598     }
599     fclose(fdat);
600     fclose(fdatTmp);
601     remove("Data/AuversackHoudart_Employes.dat");
602     rename("Data/AuversackHoudart_Employes.tmp",
        "Data/AuversackHoudart_Employes.dat");
603 }
604 void modifierFonction(int numEmploye, char newFonction[]) { //modifie la fonction
    d'un employé
605     int n=0, i;
606     FILE *fdat, *fdatTmp;
607     fdat = fopen("Data/AuversackHoudart_Employes.dat", "r");
608     fdatTmp = fopen("Data/AuversackHoudart_Employes.tmp", "w");
609     Employe *deb, *courant, *suitant;
610     courant=malloc(sizeof(Employe));
611     deb=courant;
612     // Lecture + Construction de ma liste chaînée
613     while(!feof(fdat)) {
614         fscanf(fdat, "%s", &courant->nom);
615         fscanf(fdat, "%d", &courant->service);
616         fscanf(fdat, "%s", &courant->fonction);
617     }

```

```

618     suivant=malloc(sizeof(Employe));
619     courant->suivant=suivant;
620     n++;
621     courant=suivant;
622 }
623 //Placer Null au suivant du dernière élément + libérer l'espace de suivant
624 courant=deb;
625 for(i=1;i<n;i++) {
626     courant=courant->suivant;
627 }
628 courant->suivant=NULL;
629 courant=deb;
630 // Ecriture
631 for(i=1;i<n;i++) {
632     if(i==n) {
633         strcpy(courant->fonction, nomFonction);
634     }
635     fprintf(fdatTmp, "%s %d %s", courant->nom, courant->service,
636     courant->fonction);
637     if(i!=n) {
638         fprintf(fdatTmp, "\n");
639     }
640     courant=courant->suivant;
641 }
642 fclose(fdat);
643 fclose(fdatTmp);
644 remove("Data/AuversackHoudart_Employes.dat");
645 rename("Data/AuversackHoudart_Employes.tmp",
646 "Data/AuversackHoudart_Employes.dat");
647 }
648 void modifierNomEmploye(int numEmploye, char nomEmploye[]) { //modifie le nom de
649 l'employé
650 int n=0, i;
651 FILE *fdat, *fdatTmp;
652 fdat = fopen("Data/AuversackHoudart_Employes.dat", "r");
653 fdatTmp = fopen("Data/AuversackHoudart_Employes.tmp", "w");
654 Employe *deb, *courant, *suivant;
655 courant=malloc(sizeof(Employe));
656 deb=courant;
657 while(!feof(fdat)) {
658     fscanf(fdat, "%s", &courant->nom);
659     fscanf(fdat, "%d", &courant->service);
660     fscanf(fdat, "%s", &courant->fonction);
661
662     suivant=malloc(sizeof(Employe));
663     courant->suivant=suivant;
664     n++;
665     courant=suivant;
666 }
667 //Placer Null au suivant du dernière élément + libérer l'espace de suivant
668 courant=deb;
669 for(i=1;i<n;i++) {
670     courant=courant->suivant;
671 }
672 courant->suivant=NULL;
673 courant=deb;
674 // Ecriture
675 for(i=1;i<n;i++) {
676     if(i==numEmploye) {
677         strcpy(courant->nom, nomEmploye);
678     }
679     fprintf(fdatTmp, "%s %d %s", courant->nom, courant->service,
680     courant->fonction);
681     if(i!=n) {
682         fprintf(fdatTmp, "\n");
683     }
684     courant=courant->suivant;
685 }
686 fclose(fdat);
687 fclose(fdatTmp);
688 remove("Data/AuversackHoudart_Employes.dat");
689 rename("Data/AuversackHoudart_Employes.tmp",
690 "Data/AuversackHoudart_Employes.dat");

```

## AuversackHoudart\_recuperation.c

```

1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <string.h>
4  #include "../Headers/AuversackHoudart_structures.h"
5  void recupTables() { //affiche la liste des tables
6      int n=0, i;
7      FILE *fdat;
8      fdat = fopen("Data/AuversackHoudart_Table.dat", "r");
9      Table *deb, *courant, *suivant;
10     courant=malloc(sizeof(Table));
11     deb=courant;
12     // Lecture + Construction de ma liste chaînée
13     while(!feof(fdat)) {
14         strcpy(courant->nomMatin, "");
15         courant->nbPersonneMatin=0;
16         courant->numMenuMatin=0;
17         strcpy(courant->nomSoir, "");
18         courant->nbPersonneSoir=0;
19         courant->numMenuSoir=0;
20         fscanf(fdat, "%d", &courant->estReserveMatin);
21         if(courant->estReserveMatin == 1) {
22             fscanf(fdat, "%s", &courant->nomMatin);
23             fscanf(fdat, "%d", &courant->nbPersonneMatin);
24             fscanf(fdat, "%d", &courant->numMenuMatin);
25         }
26         fscanf(fdat, "%d", &courant->estReserveSoir);
27         if(courant->estReserveSoir == 1) {
28             fscanf(fdat, "%s", &courant->nomSoir);
29             fscanf(fdat, "%d", &courant->nbPersonneSoir);
30             fscanf(fdat, "%d", &courant->numMenuSoir);
31         }
32         fscanf(fdat, "%d", &courant->nbPlaceMax);
33         suivant=malloc(sizeof(Table));
34         courant->suivant=suivant;
35         n++;
36         courant=suivant;
37     }
38     //Placer Null au suivant du dernière élément + libérer l'espace de suivant
39     courant=deb;
40     for(i=1; i<n; i++) {
41         courant=courant->suivant;
42     }
43     courant->suivant=NULL;
44     courant=deb;
45     // Affichage
46     printf("\n");
47     printf("      |Nid|\n");
48     printf("----->\n");
49     printf("      | Num table | nb place max | Reserve | Nom | nb Pers. |");
50     printf("\n");
51     printf("----->\n");
52     if(compterTable()!=0) {
53         for(i=1; i<n; i++) {
54             printf("      | %d | %d | %d | %s | %d | %d |",
55                 courant->nbPlaceMax,
56                 {courant->estReserveMatin==1 ? "Oui":"Non"}, courant->nomMatin,
57                 courant->nbPersonneMatin, courant->numMenuMatin);
58             printf("----->\n");
59             printf("      | %d | %d | %d | %s | %d | %d |",
60                 courant->nbPlaceMax,
61                 {courant->estReserveSoir==1 ? "Oui":"Non"}, courant->nomSoir,
62                 courant->nbPersonneSoir, courant->numMenuSoir);
63             printf("----->\n");
64         }
65     }
66     printf("\n");
67     courant=deb;
68     // Affichage
69     printf("      |Soir|\n");
70     printf("----->\n");

```



```

63     printf(" | Num table | nb place max | Reserve | Nom | nb Pers. |  

64     Sum menu |\n");  

65     printf("-----\n");  

66     if(compterTable()!=0) {  

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

68             printf(" | %d | %2d | %3s | %12s |\n",i,courant->nbPlaceMax,  

69                 {courant->estReserveSoir==1} ? "Oui":"Non", courant->nomSoir,  

70                 courant->nbPersonneSoir, courant->sumMenuSoir);  

71             courant=courant->suivant;  

72         }  

73         printf("-----\n");  

74     }  

75     printf("\n");  

76     free(courant);  

77     free(suivant);  

78     fclose(fdat);  

79 }  

80 void recupMenu() { //affiche la liste des menus  

81     int n=0, i, j;  

82     FILE *fdat;  

83     fdat = fopen("Data/AuversackBoudart_Menu.dat", "r");  

84     Menu *deb, *courant, *suivant;  

85     courant=malloc(sizeof(Menu));  

86     deb=courant;  

87     // Lecture + Construction de ma liste chaînée  

88     while(!feof(fdat)) {  

89         strcpy(courant->nom,"");  

90         courant->prix=0;  

91         strcpy(courant->description,"");  

92         fscanf(fdat,"%s",&courant->nom);  

93         fscanf(fdat,"%i",&courant->prix);  

94         fscanf(fdat,"%s",&courant->description);  

95         suivant=malloc(sizeof(Menu));  

96         courant->suivant=suivant;  

97         n++;  

98         courant=suivant;  

99     }  

100     //Placer Null au suivant du dernière élément + libérer l'espace de suivant  

101     courant=deb;  

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

103         courant=courant->suivant;  

104     }  

105     courant->suivant=NULL;  

106     courant=deb;  

107     // Affichage  

108     printf("\n");  

109     printf("-----\n");  

110     printf(" | N. | Nom | Prix (Euro) | Description |\n");  

111     printf("-----\n");  

112     if(compterMenu()!=0) {  

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

114             for(j=0; j<strlen(courant->nom); j++) {  

115                 if(courant->nom[j] == ' ') {  

116                     courant->nom[j] = ' ';  

117                 }  

118             }  

119             for(j=0; j<strlen(courant->description); j++) {  

120                 if(courant->description[j] == ' ') {  

121                     courant->description[j] = ' ';  

122                 }  

123             }  

124         }  

125     }  

126 }

```

```

122         printf(" | %d | %-20s | %5.2f | %s\n",i,
123             courant->nom, courant->prix, courant->description);
124         courant=courant->suivant;
125     }
126     printf("\n");
127 }
128 else {
129     printf("\n");
130 }
131 free(courant);
132 free(suivant);
133 fclose(fdat);
134 }
135 void recupEmployes() { //affiche la liste des employes
136     int n=0, i, j;
137     FILE *fdat;
138     fdat = fopen("Data/AuversackRoadart_Employes.dat", "r");
139     Employee *deb, *courant, *suivant;
140     courant=malloc(sizeof(Employee));
141     deb=courant;
142     // Lecture + Construction de ma liste chainée
143     while(!feof(fdat)) {
144         strcpy(courant->nom,"");
145         courant->service=0;
146         strcpy(courant->fonction,"");
147         fscanf(fdat,"%s",&courant->nom);
148         fscanf(fdat,"%d",&courant->service);
149         fscanf(fdat,"%s",&courant->fonction);
150         suivant=malloc(sizeof(Employee));
151         courant->suivant=suivant;
152         n++;
153         courant=suivant;
154     }
155     //Placer Null au suivant du dernière élément + libérer l'espace de suivant
156     courant=deb;
157     for(i=0;i<n;i++) {
158         courant=courant->suivant;
159     }
160     courant->suivant=NULL;
161     courant=deb;
162     // Affichage
163     printf("\n");
164     printf("    Service - Midi : 12h00-13h00 - Soir : 18h00-23h00");
165     printf("\n");
166     printf("\n");
167     printf(" | Employe |\n");
168     printf("-----\n");
169     printf(" | N |      Nom      |   Service   |   Fonction   |\n");
170     printf("-----\n");
171     if(compterEmploye()!=0) {
172         for(i=0;i<n;i++) {
173             for(j=0; j<strlen(courant->fonction); j++) {
174                 if(courant->fonction[j] == ' ') {
175                     courant->fonction[j] = '-';
176                 }
177             }
178             printf(" | %d | %s | %s | %s\n",i, courant->nom,
179                 {courant->service==1?"Midi":{courant->service==2?"Soir":"Aucun"},
180                 courant->fonction);
181             courant=courant->suivant;
182         }
183         printf("-----\n");
184     }
185     printf("\n");
186     free(courant);
187     free(suivant);
188     fclose(fdat);
189 }
190 void recupAdditionMidi() { //affiche l'addition d'une table pour le service du
midi

```



```

189 int n=0, i;
190 FILE *fdat;
191 fdat = fopen("Data/AuversschBoudart_Table.dat", "r");
192 Table *deb, *courant, *suivant;
193 courant=malloc(sizeof(Table));
194 deb=courant;
195 // Lecture + Construction de ma liste chaînée
196 while(!feof(fdat)) {
197     // Initialisation
198     courant->nbPlaceMax=0;
199     strcpy(courant->nomMatin, "");
200     courant->nbPersonneMatin = 0;
201     courant->numMenuMatin = 0;
202     strcpy(courant->nomSoir, "");
203     courant->nbPersonneSoir = 0;
204     courant->numMenuSoir = 0;
205     fscanf(fdat, "%d", &courant->estReserveMatin);
206     if(courant->estReserveMatin == 1) {
207         fscanf(fdat, "%s", &courant->nomMatin);
208         fscanf(fdat, "%d", &courant->nbPersonneMatin);
209         fscanf(fdat, "%d", &courant->numMenuMatin);
210     }
211     fscanf(fdat, "%d", &courant->estReserveSoir);
212     if(courant->estReserveSoir == 1) {
213         fscanf(fdat, "%s", &courant->nomSoir);
214         fscanf(fdat, "%d", &courant->nbPersonneSoir);
215         fscanf(fdat, "%d", &courant->numMenuSoir);
216     }
217     fscanf(fdat, "%d", &courant->nbPlaceMax);
218     suivant=malloc(sizeof(Table));
219     courant->suivant=suivant;
220     n++;
221     courant=suivant;
222 }
223 //Placer Null au suivant du dernière élément + libérer l'espace de suivant
224 courant=deb;
225 for(i=1;i<n;i++) {
226     courant=courant->suivant;
227 }
228 courant->suivant=NULL;
229 courant=deb;
230 // Affichage
231 printf("\n");
232 printf(" |Midl\\n");
233 printf("
-----\\n
234 printf(" | Num table | nb place max | Reserve | Nom | nb Pers. |
Num menu \\n");
235 printf("
-----\\n
236 ");
237 for(i=1;i<n;i++) {
238     printf(" | %d | %2d | %3s | %12s | %d
| %d |\\n", i, courant->nbPlaceMax, (courant->estReserveMatin==1 ?
"Oui":"Non"), courant->nomMatin, courant->nbPersonneMatin,
courant->numMenuMatin);
239     courant=courant->suivant;
240 }
241 printf("
-----\\n
242 ");
243 printf("\\n");
244 free(courant);
245 free(suivant);
246 fclose(fdat);
247 }
248 void recupAdditionSoir() { //affiche l'addition d'une table du service du soir
249 int n=0, i;
250 FILE *fdat;
251 fdat = fopen("Data/AuversschBoudart_Table.dat", "r");
252 Table *deb, *courant, *suivant;
253 courant=malloc(sizeof(Table));

```

```

252 deb=courant;
253 // Lecture + Construction de sa liste chaînée
254 while(!feof(fdat)) {
255     courant->nbPlaceMax=0;
256     strcpy(courant->nomMatin, "");
257     courant->nbPersonneMatin = 0;
258     courant->numMenuMatin = 0;
259     strcpy(courant->nomSoir, "");
260     courant->nbPersonneSoir = 0;
261     courant->numMenuSoir = 0;
262     fscanf(fdat,"%d",&courant->estReserveMatin);
263     if(courant->estReserveMatin == 1) {
264         fscanf(fdat,"%s",&courant->nomMatin);
265         fscanf(fdat,"%d",&courant->nbPersonneMatin);
266         fscanf(fdat,"%d",&courant->numMenuMatin);
267     }
268     fscanf(fdat,"%d",&courant->estReserveSoir);
269     if(courant->estReserveSoir == 1) {
270         fscanf(fdat,"%s",&courant->nomSoir);
271         fscanf(fdat,"%d",&courant->nbPersonneSoir);
272         fscanf(fdat,"%d",&courant->numMenuSoir);
273     }
274     fscanf(fdat,"%d",&courant->nbPlaceMax);
275     suivant=malloc(sizeof(Table));
276     courant->suivant=suivant;
277     n++;
278     courant=suivant;
279 }
280 //Placer Null au suivant du dernière élément + libérer l'espace de suivant
281 courant=deb;
282 for(i=1;i<n;i++) {
283     courant=courant->suivant;
284 }
285 courant->suivant=NULL;
286 courant=deb;
287 // Affichage
288 printf("    Soir\n");
289 printf("
-----\n
");
290 printf("    Num table | nb place max | Réservé |    Nom    | nb Pers. |
Num menu |n");
291 printf("
-----\n
");
292 for(i=1;i<n;i++) {
293     printf("    %d      | %2d      | %3s    | %12s    | %d
| %d      |n",i,courant->nbPlaceMax, {courant->estReserveSoir==1} ?
"Oui":"Non", courant->nomSoir, courant->nbPersonneSoir, courant->numMenuSoir);
    courant=courant->suivant;
294 }
295 printf("
-----\n
");
296 printf("\n");
297 free(courant);
298 free(suivant);
299 fclose(fdat);
300 }
301
302 void recupTableReserveMidi() { //affiche les tables reservees pour le midi
303     int n=0, i;
304     FILE *fdat;
305     fdat = fopen("Data/AuversackBoudart_Table.dat", "r");
306     Table *deb, *courant, *suivant;
307     courant=malloc(sizeof(Table));
308     deb=courant;
309     // Lecture + Construction de sa liste chaînée
310     while(!feof(fdat)) {
311         courant->nbPlaceMax=0;
312         strcpy(courant->nomMatin, "");
313         courant->nbPersonneMatin = 0;
314         courant->numMenuMatin = 0;
315         strcpy(courant->nomSoir, "");

```

```

316     courant->nbPersonneSoir = 0;
317     courant->numMenuSoir = 0;
318     fscanf(fdat, "%d", &courant->estReserveMatin);
319     if(courant->estReserveMatin == 1) {
320         fscanf(fdat, "%s", &courant->nomMatin);
321         fscanf(fdat, "%d", &courant->nbPersonneMatin);
322         fscanf(fdat, "%d", &courant->numMenuMatin);
323     }
324     fscanf(fdat, "%d", &courant->estReserveSoir);
325     if(courant->estReserveSoir == 1) {
326         fscanf(fdat, "%s", &courant->nomSoir);
327         fscanf(fdat, "%d", &courant->nbPersonneSoir);
328         fscanf(fdat, "%d", &courant->numMenuSoir);
329     }
330     fscanf(fdat, "%d", &courant->nbPlaceMax);
331     suivant=malloc(sizeof(Table));
332     courant->suivant=suivant;
333     n++;
334     courant=suivant;
335 }
336 //Placer Null au suivant du dernière élément + libérer l'espace de suivant
337 courant=deb;
338 for(i=1;i<=n;i++) {
339     courant=courant->suivant;
340 }
341 courant->suivant=NULL;
342 courant=deb;
343 // Affichage
344 printf("\n");
345 printf("      |Hidi|\n");
346 printf("
-----\n
");
347 printf("      | Num table | nb place max |  Reserve |      Nom      | nb Pers. |
Num menu |\n");
348 printf("
-----\n
");
349 if(compterTableReservMatin()!=0) {
350     for(i=1;i<=n;i++) {
351         if(courant->estReserveMatin==1) {
352             printf("      | %d |      | %2d |      | %-15s | %-12s|
      | %d | %d | |\n", i, courant->nbPlaceMax,
[courant->estReserveMatin==1 ? "Oui":"Non"], courant->nomMatin,
courant->nbPersonneMatin, courant->numMenuMatin);
353         }
354         courant=courant->suivant;
355     }
356     printf("
-----\n
");
357 }
358 printf("\n");
359 free(courant);
360 free(suivant);
361 fclose(fdat);
362 }
363 void recupTableReserveSoir() { //affiche les tables reservees pour le soir
364     int n=0, i;
365     FILE *fdat;
366     fdat = fopen("Data/AuversockHoodart_Table.dat", "r");
367     Table *deb, *courant, *suivant;
368     courant=malloc(sizeof(Table));
369     deb=courant;
370     // Lecture + Construction de ma liste chainée
371     while(!feof(fdat)) {
372         courant->nbPlaceMax=0;
373         strcpy(courant->nomMatin, "");
374         courant->nbPersonneMatin = 0;
375         courant->numMenuMatin = 0;
376         strcpy(courant->nomSoir, "");
377         courant->nbPersonneSoir = 0;
378         courant->numMenuSoir = 0;

```

```

379     fscanf(fdat, "%d", &courant->estReserveMatin);
380     if(courant->estReserveMatin == 1) {
381         fscanf(fdat, "%s", &courant->nomMatin);
382         fscanf(fdat, "%d", &courant->nbPersonneMatin);
383         fscanf(fdat, "%d", &courant->numMenuMatin);
384     }
385     fscanf(fdat, "%d", &courant->estReserveSoir);
386     if(courant->estReserveSoir == 1) {
387         fscanf(fdat, "%s", &courant->nomSoir);
388         fscanf(fdat, "%d", &courant->nbPersonneSoir);
389         fscanf(fdat, "%d", &courant->numMenuSoir);
390     }
391     fscanf(fdat, "%d", &courant->nbPlaceMax);
392
393     suivant=malloc(sizeof(Table));
394     courant->suivant=suivant;
395     n++;
396     courant=suivant;
397 }
398 //Placer Null au suivant du dernière élément + libérer l'espace de suivant
399 courant=deb;
400 for(i=0; i<n; i++) {
401     courant=courant->suivant;
402 }
403 courant->suivant=NULL;
404 courant=deb;
405 // Affichage
406 printf("\n");
407 printf("    | Soir\n");
408 printf("-----\n");
409 printf("    | Num table | nb.place max | Reserve |    Nom    | nb.Pers. |
410 printf("-----\n");
411 printf("-----\n");
412 if(courantTableReservSoir()!=0) {
413     for(i=0; i<n; i++) {
414         if(courant->estReserveSoir==1) {
415             printf("    | %d | %d | %d | %s | %d\n", i, courant->nbPlaceMax,
416                 (courant->estReserveSoir==1) ? "Oui" : "Non", courant->nomSoir,
417                 courant->nbPersonneSoir, courant->numMenuSoir);
418         }
419         courant=courant->suivant;
420     }
421     printf("-----\n");
422 }
423 printf("\n");
424 free(courant);
425 free(suivant);
426 fclose(fdat);
427 }
428 void recupTableLibreMidi() { //affiche les tables libres du midi
429     int n=0, i;
430     FILE *fdat;
431     fdat = fopen("Data/AuversockHoodart_Table.dat", "r");
432     Table *deb, *courant, *suivant;
433     courant=malloc(sizeof(Table));
434     deb=courant;
435     // Lecture + Construction de la liste chaînée
436     while(!feof(fdat)) {
437         courant->nbPlaceMax=0;
438         strcpy(courant->nomMatin, "");
439         courant->nbPersonneMatin = 0;
440         courant->numMenuMatin = 0;
441         strcpy(courant->nomSoir, "");
442         courant->nbPersonneSoir = 0;
443         courant->numMenuSoir = 0;

```



```

442 fscanf(fdat, "%d", &courant->estReserveMatin);
443 if(courant->estReserveMatin == 1) {
444     fscanf(fdat, "%s", &courant->nomMatin);
445     fscanf(fdat, "%d", &courant->nbPersonneMatin);
446     fscanf(fdat, "%d", &courant->numMenuMatin);
447 }
448 fscanf(fdat, "%d", &courant->estReserveSoir);
449 if(courant->estReserveSoir == 1) {
450     fscanf(fdat, "%s", &courant->nomSoir);
451     fscanf(fdat, "%d", &courant->nbPersonneSoir);
452     fscanf(fdat, "%d", &courant->numMenuSoir);
453 }
454 fscanf(fdat, "%d", &courant->nbPlaceMax);
455 suivant=malloc(sizeof(Table));
456 courant->suivant=suivant;
457 n++;
458 courant=suivant;
459 }
460 //Placer Null au suivant du dernière élément + libérer l'espace de suivant
461 courant=deb;
462 for(i=1; i<n; i++) {
463     courant=courant->suivant;
464 }
465 courant->suivant=NULL;
466 courant=deb;
467 // Affichage
468 printf("\n");
469 printf("  |Midi|\n");
470 printf("
-----~n
");
471 printf("  | Num table | nb place max | Reserve |      Nom      | nb Pers. |
Num menu |\n");
472 printf("
-----~n
");
473 if(compterTable()!=0) {
474     for(i=1; i<n; i++) {
475         if(courant->estReserveMatin==0) {
476             printf("  | %d | %d | %d | %s | %d | %d |
  | %d | %d | |\n", i, courant->nbPlaceMax,
{courant->estReserveMatin== ? "Oui": "Non"}, courant->nomMatin,
courant->nbPersonneMatin, courant->numMenuMatin);
477         }
478         courant=courant->suivant;
479     }
480     printf("
-----~n
");
481 }
482 printf("\n");
483 free(courant);
484 free(suivant);
485 fclose(fdat);
486 }
487 void recupTableLibreSoir() { //affiche les tables libres du soir
488     int n=0, i;
489     FILE *fdat;
490     fdat = fopen("Data/AuversackHoodart_Table.dat", "r");
491     Table *deb, *courant, *suivant;
492     courant=malloc(sizeof(Table));
493     deb=courant;
494     // Lecture + Construction de ma liste chaînée
495     while(!feof(fdat)) {
496         courant->nbPlaceMax=0;
497         strcpy(courant->nomMatin, "");
498         courant->nbPersonneMatin = 0;
499         courant->numMenuMatin = 0;
500
501         strcpy(courant->nomSoir, "");
502         courant->nbPersonneSoir = 0;
503         courant->numMenuSoir = 0;
504         fscanf(fdat, "%d", &courant->estReserveMatin);

```

```

505     if(courant->estReserveMatin == 1) {
506         fscanf(fdat, "%s", &courant->nomMatin);
507         fscanf(fdat, "%d", &courant->nbPersonneMatin);
508         fscanf(fdat, "%d", &courant->numMenuMatin);
509     }
510     fscanf(fdat, "%d", &courant->estReserveSoir);
511     if(courant->estReserveSoir == 1) {
512         fscanf(fdat, "%s", &courant->nomSoir);
513         fscanf(fdat, "%d", &courant->nbPersonneSoir);
514         fscanf(fdat, "%d", &courant->numMenuSoir);
515     }
516     fscanf(fdat, "%d", &courant->nbPlaceMax);
517     suivant=malloc(sizeof(Table));
518     courant->suivant=suivant;
519     n++;
520     courant=suivant;
521 }
522 //Placer Null au suivant du dernière élément + libérer l'espace de suivant
523 courant=deb;
524 for(i=0; i<n; i++) {
525     courant=courant->suivant;
526 }
527 courant->suivant=NULL;
528 courant=deb;
529 // Affichage
530 printf("\n");
531 printf("    | Soir |\n");
532 printf("-----\n");
533 printf("    | Num table | nb place max | Reserve | Nom | nb Pers. |
534 Num menu |\n");
535 printf("-----\n");
536 if(compterTable()!=0) {
537     for(i=0; i<n; i++) {
538         if(courant->estReserveSoir==0) {
539             printf("    | %d | %d | %2d | %3s | %12s |
540             %d | %d |\n", i, courant->nbPlaceMax,
541             (courant->estReserveSoir==1) ? "Oui": "Non", courant->nomSoir,
542             courant->nbPersonneSoir, courant->numMenuSoir);
543             courant=courant->suivant;
544         }
545     }
546     printf("
547     --\n");
548 }
549 printf("\n");
550 free(courant);
551 free(suivant);
552 fclose(fdat);
553 }

```



AuversackHoudart\_suppression.c

```

1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <string.h>
4  #include "../Headers/AuversackHoudart_structures.h"
5  void supprimerTable() { //supprime une table de la liste de tables
6      int n=0, i, nTable=1, tableEntree;
7      FILE *fdat, *fdatTmp;
8      fdat = fopen("Data/AuversackHoudart_Table.dat", "r");
9      fdatTmp = fopen("Data/AuversackHoudart_Table.tmp", "w");
10     afficherTablesSimple();
11     recupTables();
12     printf("  Supprimer une table : ");
13     scanf("%d", &tableEntree);
14     Table *deb, *courant, *suivant;
15     courant=malloc(sizeof(Table));
16     deb=courant;
17     // Lecture + Construction de ma liste chaînée
18     while(!feof(fdat)) {
19         fscanf(fdat, "%d", &courant->estReserveMatin);
20         if(courant->estReserveMatin == 1) {
21             fscanf(fdat, "%s", &courant->nomMatin);
22             fscanf(fdat, "%d", &courant->nbPersonneMatin);
23             fscanf(fdat, "%d", &courant->numMenuMatin);
24         }
25
26         fscanf(fdat, "%d", &courant->estReserveSoir);
27         if(courant->estReserveSoir == 1) {
28             fscanf(fdat, "%s", &courant->nomSoir);
29             fscanf(fdat, "%d", &courant->nbPersonneSoir);
30             fscanf(fdat, "%d", &courant->numMenuSoir);
31         }
32         fscanf(fdat, "%d", &courant->nbPlaceMax);
33         if(tableEntree != nTable) {
34             suivant=malloc(sizeof(Table));
35             courant->suivant=suivant;
36             n++;
37             courant=suivant;
38         }
39         nTable++;
40     }
41     //Placer Null au suivant du dernière élément + libérer l'espace de suivant
42     courant=deb;
43     for(i=1; i<n; i++) {
44         courant=courant->suivant;
45     }
46     courant->suivant=NULL;
47     // Ecriture
48     if(compterTable()!=0) {
49         courant=deb;
50         for(i=1; i<n; i++) {
51             fprintf(fdatTmp, "%d ", courant->estReserveMatin);
52             if(courant->estReserveMatin==1) {
53                 fprintf(fdatTmp, "%s %d %d ", courant->nomMatin,
54                     courant->nbPersonneMatin, courant->numMenuMatin);
55             }
56             fprintf(fdatTmp, "%d", courant->estReserveSoir);
57             if(courant->estReserveSoir==1) {
58                 fprintf(fdatTmp, "%s %d %d", courant->nomSoir,
59                     courant->nbPersonneSoir, courant->numMenuSoir);
60             }
61             fprintf(fdatTmp, "%d", courant->nbPlaceMax);
62             if(i!=n) {
63                 fprintf(fdatTmp, "\n");
64             }
65             courant=courant->suivant;
66         }
67     }
68     fclose(fdat);
69     fclose(fdatTmp);
70     if(compterTable()!=0) {
71         remove("Data/AuversackHoudart_Table.dat");
72         rename("Data/AuversackHoudart_Table.tmp", "Data/AuversackHoudart_Table.dat");
73     }
74 }

```

```

72 }
73 void supprimerMenu() { //Supprime un menu de la liste des menus
74     int n=0, i, nMenu=1, menuEntre;
75     FILE *fdat, *fdatTmp;
76     fdat = fopen("Data/AuversackHoudart_Menu.dat", "r");
77     fdatTmp = fopen("Data/AuversackHoudart_Menu.tmp", "w");
78     afficherMenuSimple();
79     recupMenu();
80     printf("    Supprimer un Menu : ");
81     scanf("%d", &menuEntre);
82     Menu *deb, *courant, *suivant;
83     courant=malloc(sizeof(Menu));
84     deb=courant;
85     // Lecture + Construction de ma liste chaînée
86     while(!feof(fdat)) {
87         fscanf(fdat, "%s", &courant->nom);
88         fscanf(fdat, "%f", &courant->prix);
89         fscanf(fdat, "%s", &courant->description);
90         if(menuEntre != nMenu) {
91             suivant=malloc(sizeof(Menu));
92             courant->suivant=suivant;
93             n++;
94             courant=suivant;
95         }
96         nMenu++;
97     }
98     //Placer Null au suivant du dernière élément + libérer l'espace de suivant
99     courant=deb;
100     for(i=1; i<n; i++) {
101         courant=courant->suivant;
102     }
103     courant->suivant=NULL;
104     courant=deb;
105     // Ecriture
106     for(i=1; i<n; i++) {
107         fprintf(fdatTmp, "%s %0.2f %s", courant->nom, courant->prix,
108             courant->description);
109         if(i!=n) {
110             fprintf(fdatTmp, "\n");
111         }
112         courant=courant->suivant;
113     }
114     fclose(fdat);
115     fclose(fdatTmp);
116     remove("Data/AuversackHoudart_Menu.dat");
117     rename("Data/AuversackHoudart_Menu.tmp", "Data/AuversackHoudart_Menu.dat");
118 }
119 void supprimerEmploye() { //supprime un employé de la liste
120     int n=0, i, nEmploye=1, employeEntre, testDigit;
121     FILE *fdat, *fdatTmp;
122     fdat = fopen("Data/AuversackHoudart_Employes.dat", "r");
123     fdatTmp = fopen("Data/AuversackHoudart_Employes.tmp", "w");
124     system("cls");
125     afficherGestionPersonnelSimple();
126     recupEmploye();
127     printf("    Supprimer un Employe : ");
128     testDigit = scanf("%d", &employeEntre);
129     Employe *deb, *courant, *suivant;
130     courant=malloc(sizeof(Employe));
131     deb=courant;
132     // Lecture + Construction de ma liste chaînée
133     while(!feof(fdat)) {
134         fscanf(fdat, "%s", &courant->nom);
135         fscanf(fdat, "%d", &courant->service);
136         fscanf(fdat, "%s", &courant->fonction);
137         if(employeEntre != nEmploye) {
138             suivant=malloc(sizeof(Employe));
139             courant->suivant=suivant;
140             n++;
141             courant=suivant;
142         }
143         nEmploye++;
144     }

```

```

144 //Placer Null au suivant du dernière élément + libérer l'espace de suivant
145 courant=déb;
146 for(i=1;i<n;i++) {
147     courant=courant->suitant;
148 }
149 courant->suitant=NULL;
150 courant=déb;
151 if(compterEmploye()!=0) {
152     // Ecriture
153     for(i=1;i<n;i++) {
154         fprintf(fdatTmp, "%s %d %s", courant->nom, courant->service,
            courant->fonction);
155         if(i!=n) {
156             fprintf(fdatTmp, "\n");
157         }
158         courant=courant->suitant;
159     }
160 }
161 fclose(fdat);
162 fclose(fdatTmp);
163 if(compterEmploye()!=0) {
164     remove("Data/AuversackHoudart_Employes.dat");
165     rename("Data/AuversackHoudart_Employes.tmp",
        "Data/AuversackHoudart_Employes.dat");
166 }
167 }
168 void supprimerReservation() { //Permet de supprimer une reservation sur une table
169     int service, numTable, modifOk=0;
170     erreurIndexService;//gestion d'erreur
171     system("cls");
172     afficherReservation();
173     recupTableReserveMidi();
174     recupTableReserveSoir();
175     printf("\n Avez-vous reserve a midi(1) ou au soir(2) ? ");
176     scanf("%d", &service);
177     if(service==1 && compterTableReservMatin()!=0) {
178         erreurIndexNumTableMidi;//gestion d'erreur
179         system("cls");
180         afficherReservation();
181         recupTableReserveMidi();
182         printf("\n Quelle table avez-vous reserve ? ");
183         scanf("%d", &numTable);
184         if(numTable<=0 || numTable > compterTable()) {
185             viderBuffer();
186             goto erreurIndexNumTableMidi;
187         }
188         modifOk = modifTableSuppRes(1, numTable);
189         if(modifOk==1) {
190             printf("\n Suppression reussie !\n\n");
191         }
192         else {
193             printf("\n La reservation n'existe pas !\n\n");
194         }
195     }
196     else if(service==2 && compterTableReservSoir()!=0) {
197         erreurIndexNumTableSoir;//gestion d'erreur
198         system("cls");
199         afficherReservation();
200         recupTableReserveSoir();
201         printf("\n Quelle table avez-vous reserve ? ");
202         scanf("%d", &numTable);
203         if(numTable<=0 || numTable > compterTable()) {
204             viderBuffer();
205             goto erreurIndexNumTableSoir;
206         }
207         modifOk = modifTableSuppRes(2, numTable);
208         if(modifOk==1) {
209             printf("\n Suppression reussie !\n\n");
210         }
211         else {
212             printf("\n La reservation n'existe pas !\n\n");
213         }
214     }
215 }

```

```
215     else{  
216         viderBuffer();  
217         goto erreurIndexService;  
218     }  
219 }
```

AuversackHoudart\_affichage.h

```
#ifndef DEF_AuversackHoudart_affichage
#define DEF_AuversackHoudart_affichage
//contient toutes les fonctions gérant l'affichage
void afficherEcranPrincipal();
void afficherGestionPersonnel();
void afficherMenu();
void afficherMenuSimple();
void afficherServices();
void afficherTables();
void afficherAddition();
void afficherAdditionMidi();
void afficherAdditionSoir();
void afficherChangerCommande();
void afficherReservation();
void afficherReservationOptions();
#endif
```

AuversackHoudart\_ajout.h

```
#ifndef DEF_AuversackHoudart_ajout
#define DEF_AuversackHoudart_ajout
//permet d'ajouter des éléments aux listes chaînées de tables, menus et employé
void ajouterTable();
void ajouterMenu();
void ajouterEmploye();
#endif
```



AuversackHoudart\_compter.h

```
#ifndef DEF_AuversackHoudart_compter
#define DEF_AuversackHoudart_compter
//permet de compter les éléments d'une liste
int compterMenu();
int compterTable();
int compterTableReservMatin();
int compterTableReservSoir();
int compterEmploye();
#endif
```

## AuversackHoudart\_operation.h

```
#ifndef DEF_AuversackHoudart_operation
#define DEF_AuversackHoudart_operation
//permet de faire différentes opérations comme une addition, une réservation, une commande ou modifier un employé
void FaireAddition(int, int);
void changerCommande();
void faireReservation();
void modifierEmploye();
#endif
```

AuversackHoudart\_outils.h

```
#ifndef DEF_AuversackHoudart_outils
#define DEF_AuversackHoudart_outils
//différentes fonctions utilisées pour simplifié d'autres fonctions
int estReserve(int , int);
const char* nonMenuChoisi(int);
void remplaceMenuTable(int , int , int);
void selectTable(int , int);
int rechercheTableLibre(int , int );
int remplaceNonReserveTable(int , char[], int , int , int );
int modifTableSuppRes(int , int );
void viderBuffer();
void modifierService(int, int);
void modifierFonction(int, char[]);
void modifierNomEmploye(int, char[]);
#endif
```

AuversackHoudart\_recuperation.h

```
#ifndef DEF_AuversackHoudart_recuperation
#define DEF_AuversackHoudart_recuperation
//permet de récupérer des listes et de les afficher
void recupTables();
void recupMenu();
void recupEmployes();
void recupAdditionMidi();
void recupAdditionSoir();
void recupTableReserveMidi();
void recupTableReserveSoir();
void recupTableLibreMidi();
void recupTableLibreSoir();
#endif
```

AuversackHoudart\_suppression.h

```
#ifndef DEF_AuversackHoudart_suppression
#define DEF_AuversackHoudart_suppression
//permet de supprimer un élément d'une liste ou une réservation
void supprimerTable();
void supprimerMenu();
void supprimerEmploye();
void supprimerReservation();
#endif
```

AuversackHoudart\_structures.h

```
#ifndef DEF_AuversackHoudart_structures
#define DEF_AuversackHoudart_structures
//récupères les différentes structures utilisées dans le code
typedef struct Table {
    int nbPlaceMax;
    int estReserveMatin;
    char nomMatin[21];
    int nbPersonneMatin;
    int numMenuMatin;
    int estReserveSoir;
    char nomSoir[21];
    int nbPersonneSoir;
    int numMenuSoir;
    struct Table *suivant;
}Table;
typedef struct Menu {
    char nom[21];
    float prix;
    char description[60];
    struct Menu *suivant;
}Menu;
typedef struct Employe {
    char nom[21];
    int service;
    char fonction[21];
    struct Employe *suivant;
}Employe;
#endif
```



AuversackHoudart\_ressources.rc

**1** **ICON** "Icones/AuversackHoudart\_01.ico"