



République Tunisienne
Université de la Mannouba
École Nationale des Sciences de l'Informatique
ENSI

Projet C : Application de gestion d'une agence de location des voitures

E-Cars

Réalisé par :

Mohamed Rayen Jomaa

Groupe II1C

Encadré par :

Dr. Hatem Aouadi

Année universitaire : **2023-2024**

Table des matières

Introduction générale.....	1
1 Les structures	2
1.1 Structure Date.....	2
1.2 Structure Voiture.....	2
1.3 Structure Location.....	3
1.4 Structure Client	3
1.5 Structure Resultat.....	4
2 Menu de l'application	4
3 Les prototypes	6
4 Realloc	7
5 Tableau dynamique d'adresses.....	8
6 Les fichiers	9
7 Les traitements	17
7.1 La gestion des voitures	17
7.2 La gestion des clients	22
Conclusion générale.....	25
Annexes.....	26

Table des figures

1	Code du structure Date.....	2
2	Code du structure Voiture	3
3	Code du structure Location	3
4	Code du structure Client	4
5	Code du structure Resultat	4
6	Menu principal	5
7	Menu de la gestion des clients.....	5
8	Menu de la gestion des voitures.....	6
9	La navigation entre les menus.	6
10	Prototypes des fonctions.....	7
11	Fonction realloc	7
12	Déclaration et allocation du tableau dynamique d'adresses.....	8
13	Remplissage du tableau résultat	8
14	Prototype du fonction du remplissage du tableau résultat	9
15	Création du fichier	10
16	Remplissage du fichier.....	10
17	Affichage du fichier.....	10
18	Lecture d'un client du fichier.....	11
19	Ecriture d'un client dans un fichier.....	12
20	Affichage du fichier clients	13
21	Affichage du fichier ClientsIndex.....	14
22	Modification d'un client à partir de sa position dans le fichier	15
23	Affichage du fichier apres modification	16
24	Code du tableau remplissage des voitures	17
25	Exécution du remplissage du tableau voitures	18
26	Modification d'une voiture.....	19
27	Exécution de modification d'une voiture	20
28	Suppression d'une voiture.....	21
29	Exécution de la suppression d'une voiture	21
30	Code de remplissage du tableau clients	22
31	Exécution du remplissage du tableau clients	22
32	Code d'affichage du tableau clients.....	23
33	Exécution de l'affichage du tableau clients	23
34	Code d'affichage du client qui a le plus grand nombre de locations	24
35	Exécution d'affichage du client qui a le plus grand nombre de locations	24

Introduction générale

Les applications de gestion jouent un rôle crucial dans la rationalisation des opérations commerciales, et dans le secteur de la location de voitures, elles deviennent un outil inestimable pour optimiser l'efficacité opérationnelle. Dans le monde en constante évolution de la mobilité, les agences de location de voitures font face à des défis complexes liés à la gestion de flotte, à la réservation, à la maintenance des véhicules, à la satisfaction client, et à la croissance économique.

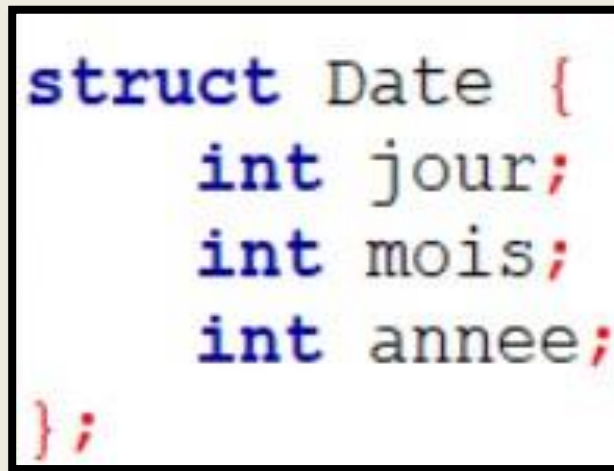
Dans ce rapport je vais présenter mon application de gestion d'une agence de location des voitures, nommée E-cars, réalisée en langage C.

1 Les structures

1.1 Structure Date

La structure Date est composée par 3 champs : jour qui est de type **int**, mois de type **int** et année de type **int**. Elle permet de stocker une date.

La figure 1 ci-jointe représente une capture d'écran du code de définition du structure Date.



```
struct Date {  
    int jour;  
    int mois;  
    int annee;  
};
```

Fig. 1 : Code du structure Date

1.2 Structure Voiture

La structure Voiture est composée par 5 champs : code qui est de type **int**, modele de type **char[50]**, marque de type **char[50]**, année de type **int** qui désigne l'année de fabrication de la voiture et prixLocation de type **float**. Elle permet de stocker une voiture. La figure 2 ci-après représente une capture d'écran du code de définition du structure Voiture.

```

struct Voiture {
    int code;
    char modele[50];
    char marque[50];
    int annee;
    float prixLocation;
};

```

Fig. 2 : Code du structure Voiture

1.3 Structure Location

La structure Location est composée par 4 champs : code qui est de type **int** qui est l'identifiant de la location (nombre aléatoire composé par 9 chiffres), voitureLouee qui est de type **VOITURE *** (un pointeur sur voiture), dateDebut de type **DATE** et dateFin de type **DATE**. Cette structure permet de stocker les détails d'une location d'une voiture donnée.

La figure 3 ci-après représente une capture d'écran du code de définition du structure Location.

```

struct Location {
    int code;
    VOITURE * voitureLouee;
    DATE dateDebut;
    DATE dateFin;
};

```

Fig. 3 : Code du structure Location

1.4 Structure Client

La structure Client est composée par 6 champs : prenom qui est de type **char[50]**, nom de type **char[50]**, téléphone de type **int**, le nombreLocations de type **int** et locations qui est de type **LOCATION *** qui renseigne les locations d'un client donné. Cette structure permet de stocker les détails d'un client donné.

La figure 4 ci-après représente une capture d'écran du code de définition du structure Client.

```

struct Client {
    int cin;
    char nom[50];
    char prenom[50];
    int telephone;
    LOCATION * locations[100];
    int nombreLocations;
};

```

Fig. 4 : Code du structure Client

1.5 Structure Resultat

La structure Resultat est composée par 3 champs : nomClient qui est de type **char[50]**, prenomClient de type **char[50]** et nbLocations de type **int**. Cette structure permet de stocker le nombre de locations pour chaque client.

La figure 5 ci-jointe désigne une capture d'écran du code de définition du structure Resultat.

```

struct Resultat {
    char nomClient[50];
    char prenomClient[50];
    int nbLocations;
};

```

Fig. 5 : Code du structure Resultat

2 Menu de l'application

Notre application est composée par un menu principal constitué par : La gestion des voitures et la gestion des clients.

La figure 6 ci-jointe représente le menu principal de notre application.

```
***** MENU PRINCIPAL *****

1 - Manipuler les clients
2 - Manipuler les voitures
3 - Quitter

Choisissez une option :
```

Fig. 6 : Menu principal

En choisissant une option, un sous menu apparaît au chef d'agence pour manipuler soit les voitures soit les clients. Ci-après, la figure 7 renseigne le menu de la gestion des clients.

```
***** MENU PRINCIPAL *****

1 - Manipuler les clients
2 - Manipuler les voitures
3 - Quitter

Choisissez une option : 1

***** MENU CLIENTS *****

1 - Afficher les clients (a partir du fichier)
2 - Ajouter des clients
3 - Modifier un client
4 - Trouver le client avec le plus de locations
5 - Afficher fichier index
6 - Revenir au menu principal

Choisissez une option :
```

Fig. 7 : Menu de la gestion des clients

Et la figure 8 illustre le menu de la gestion des voitures.

```
***** MENU PRINCIPAL *****

1 - Manipuler les clients
2 - Manipuler les voitures
3 - Quitter

Choisissez une option : 2

***** MENU VOITURES *****

1 - Afficher les voitures
2 - Ajouter une voiture
3 - Supprimer une voiture
4 - Mettre à jour une voiture
5 - Revenir au menu principal
Choisissez une option :
```

Fig. 8 : Menu de la gestion des voitures

Notez bien : Il y'a une navigation entre les menus ; si le chef d'agence est dans le menu de la gestion des clients il peut revenir au menu principal comme le montre la figure 9.

```
1 - Manipuler les clients
2 - Manipuler les voitures
3 - Quitter

Choisissez une option : 2

***** MENU VOITURES *****

1 - Afficher les voitures
2 - Ajouter une voiture
3 - Supprimer une voiture
4 - Mettre à jour une voiture
5 - Revenir au menu principal
Choisissez une option : 5
Retour au menu principal...

***** MENU PRINCIPAL *****

1 - Manipuler les clients
2 - Manipuler les voitures
3 - Quitter

Choisissez une option :
```

Fig. 9 : La navigation entre les menus.

3 Les prototypes

La figure 10 ci-après illustre les prototypes des fonctions de notre application.

```

void ajouterVoiture (VOITURE *); ///Créer une nouvelle voiture
void afficherVoiture (VOITURE); ///Afficher les informations d'une voiture
void ajouterClient (CLIENT *, VOITURE *, int); ///Créer un nouveau client
void afficherClient (CLIENT); ///Afficher les informations d'une voiture
void affecterClientToVoiture (CLIENT *, VOITURE); ///affecter une voiture donnée à un client précis
VOITURE rechercherVoitureParCode(int, VOITURE [], int); ///Chercher une voiture par son code et la retourner
void miseAJour (VOITURE *, int, int); ///mettre à jour une voiture
void supprimerVoiture (VOITURE *, int, int); ///supprimer une voiture
void afficherTabVoiture (VOITURE *, int); ///Afficher le tableau des voitures
void affichetabclient (CLIENT *, int); ///Afficher le tableau des clients
CLIENT* allocationclient (int); ///Allouer le tableau client et retourner l'adresse du 1ere case
VOITURE* allocationvoiture (int); ///allocation du tableau voiture et retourner l'adresse du 1ere case
void remplirTabVoiture (VOITURE *, int); ///Remplir le tableau voiture
void reallocVoiture (VOITURE *, int, int); ///redimensionner le tableau voiture
void remplirTabClient (CLIENT *, int, VOITURE *, int); ///Remplir le tableau client
void remplirTabResultat (RESULTAT **, int, CLIENT *, int, int); ///Remplir le tableau resultats
void maxNbLocations (RESULTAT **, int); ///Rechercher et afficher le client qui a le plus grand nb de locations
void creerFichierClientsIndex (FILE **, FILE **); ///Créer fichier structure clients et fichier index
void remplirFichierClientsIndex (FILE *, FILE *, VOITURE *, int, CLIENT *, int); ///Remplir fichier index par des clients
void afficherFichierClientsIndex (FILE *, FILE *, int); ///Afficher le contenu du fichier index
void ecrireUnClientDansFichier (FILE *, CLIENT);
CLIENT lireUnClientDuFichier (FILE *);
#endif // BIB_H_INCLUDED

```

Fig. 10 : Prototypes des fonctions

4 Realloc

La fonction **realloc** en langage C est utilisée pour modifier la taille de la mémoire allouée dynamiquement précédemment par malloc, calloc, ou realloc lui-même. Elle permet de réallouer de la mémoire pour un bloc de mémoire déjà alloué, en modifiant sa taille.

Dans notre cas, nous avons utilisé **realloc** pour redimensionner le tableau des voitures. Si le chef d'agence veut ajouter des voitures supplémentaires en plus de sa base de données, il peut le faire grâce à la fonction realloc.

Ci-dessous, nous avons présenter dans la figure 11 la capture d'écran de la fonction **reallocVoiture**.

```

void reallocVoiture (VOITURE **v, int *nbVoitures, int nbNouveauxVoitures) {
    // Redimensionner le tableau de voitures v
    *v = (VOITURE *)realloc(*v, (*nbVoitures + nbNouveauxVoitures) * sizeof (VOITURE));

    if (*v == NULL) {
        // Si la réallocation a échoué
        printf("Erreur lors du redimensionnement du tableau stockvoiture\n");
    } else {
        // Modifier la taille du tableau voitures v initiale
        *nbVoitures += nbNouveauxVoitures;
        // Ajouter de nouvelles voitures
        int i;
        for (i = *nbVoitures - nbNouveauxVoitures; i < *nbVoitures; i++) {
            printf("\nVOITURE %d\n", i);
            ajouterVoiture(*v + i);
        }
    }
}

```

Fig. 11 : Fonction realloc

5 Tableau dynamique d'adresses

Lors du développement de mon application, j'ai recours à utiliser un tableau dynamique d'adresses de structure RESULTAT.

J'ai déclaré le tableau et allouer de l'espace mémoire de cette façon comme le montre la figure 14.

```
RESULTAT ** tableauResultats;  
// Nombre initial d'éléments dans le tableau  
int nbResultats = 0;  
// Capacité initiale du tableau  
int capaciteResultats = 5;  
// Allocation du tableau dynamique d'adresses  
tableauResultats = (RESULTAT **)malloc(capaciteResultats * sizeof(RESULTAT *));
```

Fig. 12 : Déclaration et allocation du tableau dynamique d'adresses

La figure ci-après illustre le remplissage du tableau dynamique `tableauResultats`.

```
printf("Vous avez choisi de trouver le client avec le plus de locations.\n");  
if (tableauResultats == NULL) {  
    printf("Erreur lors de l'allocation du tableau de résultats\n");  
    exit(-1);  
}  
//remplissage du tableau dynamique d'adresses  
remplirTabResultat(tableauResultats, nbResultats, clientsAgence, nbClients, capaciteResultats)  
// Trouver le client avec le montant le plus élevé à payer  
maxNbLocations(tableauResultats, nbClients);  
// Libérer la mémoire allouée pour chaque résultat  
for (i = 0; i < nbResultats; i++) {  
    free(tableauResultats[i]);  
}
```

Fig. 13 : Remplissage du tableau résultat

```

void remplirTabResultat (RESULTAT ** tableauResultats, int nbResultats, CLIENT *c, int nbClients
                        , int capaciteResultats){
    int j;
    RESULTAT *nouveauResultat;
    for (j=0; j<nbClients; j++){
        nouveauResultat = (RESULTAT *)malloc(sizeof(RESULTAT));
        if (nouveauResultat == NULL) {
            printf("Erreur lors de l'allocation d'un nouveau résultat\n");
            exit(-2);
        }
        strcpy(nouveauResultat->nomClient, (c+j)->nom);
        strcpy(nouveauResultat->prenomClient, (c+j)->prenom);
        nouveauResultat->nbLocations = (c+j)->nombreLocations;
        int prix=0, v;
        // Vérifier si le tableau a besoin d'être redimensionné
        if (nbResultats >= capaciteResultats) {
            capaciteResultats *= 2; // Double la capacité du tableau
            tableauResultats = (RESULTAT **)realloc(tableauResultats,
                                                    capaciteResultats * sizeof(RESULTAT *));

            if (tableauResultats == NULL) {
                if (tableauResultats == NULL) {
                    printf("Erreur lors du redimensionnement du tableau de résultats\n");
                    exit(-3);
                }
            }
        }

        // Ajouter l'adresse du nouveau résultat dans le tableau
        *(tableauResultats+nbResultats) = nouveauResultat;
        nbResultats++; // Augmenter le nombre de résultats
    }
}

```

Fig. 14 : Prototype de fonction du remplissage du tableau résultat

6 Les fichiers

J'ai utilisé deux fichiers binaires ; fichier Clients et fichier ClientsIndex. Le fichier Clients stocke les données des clients de l'agence alors que ClientsIndex stocke la position du curseur dans le fichier Clients. Les figures ci-après représente les fonctions de gestion des fichiers.

```

void creerFichierClientsIndex(FILE**fp, FILE**fi)
{
    *fp=fopen("Clients", "wb+");
    if( !*fp) exit(-1);
    *fi=fopen("ClientsIndex", "wb+");
    if ( !*fi) exit(-1);
}

```

Fig. 15 : Création du fichier

```

void remplirFichierClientsIndex(FILE*fp, FILE*fi, VOITURE*v, int nbVoitures, CLIENT * c, int nbClients)
{
    int x;
    int i;
    for (i=0; i<nbClients; i++){
        ajouterClient(c+i, v, nbVoitures);
        x=ftell(fp);
        fwrite(&x, sizeof(int), 1, fi);
        ecrireUnClientDansFichier(fp, *(c+i));
    }
}

```

Fig. 16 : Remplissage du fichier

```

void afficherFichierClientsIndex(FILE*fp, FILE*fi, int nbClients)
{
    CLIENT c;
    int x, i;
    rewind(fi); rewind(fp);
    printf("\nAffichage Fichier index\n");

    while (fread(&x, sizeof(int), 1, fi) == 1) {
        fseek(fp, x, SEEK_SET); // Se positionner à l'offset x dans le fichier clients

        c = lireUnClientDuFichier(fp);
        afficherClient(c);
    }
}

```

Fig. 17 : Affichage du fichier

```

// Fonction pour lire les données d'un client depuis un fichier
CLIENT lireUnClientDuFichier(FILE *fp) {
    CLIENT c;
    fread(&c.cin, sizeof(int), 1, fp);
    fread(c.nom, sizeof(char), 50, fp);
    fread(c.prenom, sizeof(char), 50, fp);
    fread(&c.telephone, sizeof(int), 1, fp);
    fread(&c.nombreLocations, sizeof(int), 1, fp);

    // Allocation de mémoire pour les locations

    if (c.locations == NULL) {
        perror("Allocation de mémoire a échoué");
        exit(-3);
    }
    int i ;
    for (i= 0; i < c.nombreLocations; ++i) {
        c.locations[i] = (LOCATION *)malloc(sizeof(LOCATION));
        if (c.locations[i] == NULL) {
            perror("Allocation de mémoire a échoué");

            exit(-3);
        }

        fread(&c.locations[i]->code, sizeof(int), 1, fp);
    }

    return c;
}

```

Fig. 18 : Lecture d'un client du fichier


```

void ecrireUnClientDansFichier(FILE*fp,CLIENT c)
{
    // Écriture des informations du client dans le fichier
    fwrite(&c.cin, sizeof(int), 1, fp);
    fwrite(c.nom, sizeof(char), 50, fp);
    fwrite(c.prenom, sizeof(char), 50, fp);
    fwrite(&c.telephone, sizeof(int), 1, fp);
    fwrite(&c.nombreLocations, sizeof(int), 1, fp);
    //fwrite(c.locations, sizeof(LOCATION), c.nombreLocations, fp);

    // Écriture des informations de chaque location
    int i;
    for ( i = 0; i < c.nombreLocations; ++i) {
        fwrite(&c.locations[i]->code, sizeof(int), 1, fp);
    }
}

```

Fig. 19 : Ecriture d'un client dans un fichier

La figure 20 représente l’affichage du contenu du fichier Clients.

```
***** MENU CLIENTS *****
1 - Afficher les clients (a partir du fichier)
2 - Ajouter des clients
3 - Modifier un client
4 - Trouver le client avec le plus de locations
5 - Afficher fichier index
6 - Revenir au menu principal
Choisissez une option : 1

*****Affichage*****

Affichage Fichier index
Le CIN du client: 14010163
Le nom et prnom du client: Jomaa Rayen
Le telephone du client : 97332857
Le nombre de ces locations: 0
Codes locations
```

Fig. 20 : Affichage du fichier clients

Ci-jointe, la figure 21 représente l’affichage du fichier ClientsIndex.

```
***** MENU CLIENTS *****

1 - Afficher les clients (a partir du fichier)
2 - Ajouter des clients
3 - Modifier un client
4 - Trouver le client avec le plus de locations
5 - Afficher fichier index
6 - Revenir au menu principal
Choisissez une option : 5
Affichage du fichier index
Affichage Fichier index

0
112
```

Fig. 21 : Affichage du fichier ClientsIndex

Grace à cette application, le chef d'agence a la main de modifier les données d'un client à partir de sa position dans le fichier comme illustre la figure 22 .

```
***** MENU CLIENTS *****
1 - Afficher les clients (a partir du fichier)
2 - Ajouter des clients
3 - Modifier un client
4 - Trouver le client avec le plus de locations
5 - Afficher fichier index
6 - Revenir au menu principal
Choisissez une option : 3
Vous avez choisi de mettre 0 jour un client.
Saisir la position du client 0 modifier 1
Client avant mise a jour
Le CIN du client: 14010163
Le nom et prénom du client: Jomaa Rayen
Le telephone du client : 97332857
Le nombre de ces locations: 0
Codes locations

Saisir les nouveaux données du client
Saisir le CIN du client 123
Saisir le nom du client Jm
Saisir le prenom du client MedRayen
Saisir le telephone du client 97332857
Saisir le nombre de locations de ce client 0
```

Fig. 22 : Modification d'un client à partir de sa position dans le fichier

```
*****Affichage*****  
  
Affichage Fichier index  
  
Client  
Le CIN du client: 123  
Le nom et prÚnom du client: BenAhmed Ahmed  
Le telephone du client : 32654789  
Le nombre de ces locations: 1  
Codes locations  
456  
-----  
  
Client  
Le CIN du client: 456  
Le nom et prÚnom du client: Salhi Housseem  
Le telephone du client : 52741852  
Le nombre de ces locations: 1  
Codes locations  
14155968  
-----
```

Fig. 23 : Affichage du fichier après modification

Med Rayen JOMAA

7 Les traitements

7.1 La gestion des voitures

- Remplissage

Ci-joint la figure 29 montre le code de remplissage du tableau voitures.

```
printf("\nBienvenue dans votre Agence de voiture\n\n");
printf("Combien de voiture voulez vous ajouter ? ");
scanf("%d",&nbVoitures);
stockvoiture = allocationvoiture(nbVoitures);
printf("\n*****Ajout des voitures*****\n");
remplirTabVoiture (stockvoiture, nbVoitures);

void remplirTabVoiture (VOITURE *v, int nbVoitures){
    int i;
    for(i=0;i<nbVoitures;i++){
        printf("\nVOITURE %d\n",i+1);
        ajouterVoiture(v+i);
    }
}

void ajouterVoiture(VOITURE *v) {
    printf("Donner le code de la voiture ");

    scanf("%d", &v->code);
    printf("Donner le modèle de la voiture ");
    scanf("%s", &v->modele);
    printf("Donner la marque de la voiture ");
    scanf("%s", &v->marque);
    printf("Donner l'année de fabrication de la voiture ");
    scanf("%d", &v->annee);
    printf("Donner le prix de location de la voiture ");
    scanf("%f", &v->prixLocation);
}
```

Fig. 24 : Code du tableau remplissage des voitures

```
Bienvenue dans votre Agence de voiture
Combien de voiture voulez vous ajouter ? 2
*****Ajout des voitures*****

VOITURE 1
Donner le code de la voiture 1
Donner le modPle de la voiture Sportege
Donner la marque de la voiture Kia
Donner l'annÚe de fabrication de la voiture 2023
Donner le prix de location de la voiture 200

VOITURE 2
Donner le code de la voiture 2
Donner le modPle de la voiture GrandI10
Donner la marque de la voiture Hyundai
Donner l'annÚe de fabrication de la voiture 2021
Donner le prix de location de la voiture 100
```

Fig. 25 : Exécution du remplissage du tableau voitures

- **Modification**

Ci-joint le code de modification d'une voiture

```
case 4:
    printf("Vous avez choisi de mettre à jour une voiture.\n");
    miseAJour(stockvoiture, nbVoitures, codeVoiture);
    break;
```

```
void miseAJour(VOITURE *tabVoitures, int nbVoitures, int codeVoiture) {
    int i;
    VOITURE nouvelleVoiture;
    printf("Veuillez entrer le code de la voiture à mettre à jour : ");
    scanf("%d", &codeVoiture);
    printf("Veuillez entrer les nouvelles informations de la voiture :\n");
    printf("Saisissez le nouveau code de la voiture : ");
    scanf("%d", &nouvelleVoiture.code);

    printf("Saisissez le modèle de la nouvelle voiture : ");
    scanf(" %s", &nouvelleVoiture.modele);

    printf("Saisissez la marque de la nouvelle voiture : ");
    scanf(" %s", &nouvelleVoiture.marque);

    printf("Saisissez l'année de fabrication de la nouvelle voiture : ");
    scanf("%d", &nouvelleVoiture.annee);

    printf("Saisissez le prix de location de la nouvelle voiture : ");
    scanf("%f", &nouvelleVoiture.prixLocation);
    for (i = 0; i < nbVoitures; ++i) {
        if ((tabVoitures+i)->code == codeVoiture) {
            *(tabVoitures+i) = nouvelleVoiture;
            int positionAModifier = i; // Position à modifier

            /// modifierStructure("fichier_structure.dat", positionAModifier, nouvelleVoiture);
            printf("La mise à jour de la voiture avec le code %d a été effectuée avec succès.\n", codeVoiture);
            return;
        }
    }
    printf("Aucune voiture avec le code %d n'a été trouvée. La mise à jour a échoué.\n", codeVoiture);
}
```

Fig. 26 : Modification d'une voiture

```

***** MENU VOITURES *****

1 - Afficher les voitures
2 - Ajouter une voiture
3 - Supprimer une voiture
4 - Mettre à jour une voiture
5 - Revenir au menu principal
Choisissez une option : 4
Vous avez choisi de mettre à jour une voiture.
Veuillez entrer le code de la voiture à mettre à jour : 1
Veuillez entrer les nouvelles informations de la voiture :
Saisissez le nouveau code de la voiture : 3
Saisissez le modèle de la nouvelle voiture : Rio
Saisissez la marque de la nouvelle voiture : Kia
Saisissez l'année de fabrication de la nouvelle voiture : 2022
Saisissez le prix de location de la nouvelle voiture : 150
La mise à jour de la voiture avec le code 1 a été effectuée avec succès.

*****Affichage des voitures*****

VOITURE 0

Affichage
Le code de la voiture : 3
Le modèle de la voiture: Rio
La marque de la voiture: Kia
L'année de fabrication de la voiture: 2022
Le prix de la voiture: 150.00

VOITURE 1

Affichage
Le code de la voiture : 2
Le modèle de la voiture: GrandI10
La marque de la voiture: Hyundai
L'année de fabrication de la voiture: 2021
Le prix de la voiture: 100.00

```

Fig. 27 : Exécution de modification d'une voiture

- **Suppression**

Ci-joint la figure 29 montre le code de suppression d'une voiture.

```
case 3:

    printf("Vous avez choisi de supprimer une voiture.\n");
    printf("Veuillez entrer le code de la voiture à supprimer : ");
    scanf("%d", &codeVoiture);
    supprimerVoiture(stockvoiture, nbVoitures, codeVoiture);
    nbVoitures--;
break;
```

```
void supprimerVoiture(VOITURE *tabVoitures, int nbVoitures, int codeVoiture) {
    int i, j;
    int voitureTrouvee = 0;

    afficherTabVoiture(tabVoitures, nbVoitures);
    for (i = 0; i < nbVoitures; i++) {
        printf("\nCompteur %d\n", i);
        printf("(tabVoitures+i)->code = %d\n", (tabVoitures+i)->code);
        printf("codeVoiture = %d\n", codeVoiture);
        if ((tabVoitures+i)->code == codeVoiture) {
            for (j = i; j < (nbVoitures - 1); j++) {
                *(tabVoitures+j) = *(tabVoitures+j+ 1);
            }

            printf("La voiture avec le code %d a été supprimée avec succès.\n", codeVoiture);
            voitureTrouvee = 1;
            break;
        }
    }

    if (!voitureTrouvee) {
        printf("Aucune voiture avec le code %d n'a été trouvée. La suppression a échoué.\n", codeVoiture);
    }
}
```

Fig. 28 : Suppression d'une voiture

```
***** MENU VOITURES *****

1 - Afficher les voitures
2 - Ajouter une voiture
3 - Supprimer une voiture
4 - Mettre Ó jour une voiture
5 - Revenir au menu principal
Choisissez une option : 3
Vous avez choisi de supprimer une voiture.
Veuillez entrer le code de la voiture Ó supprimer : 1

VOITURE 0

Affichage
Le code de la voiture : 1
Le modble de la voiture: aa
La marque de la voiture: aa
L'annÚe de fabrication de la voiture: 212
Le prix de la voiture: 21.00
La voiture avec le code 1 a ÚtÚ supprimÚe avec succbs.
```

Fig. 29 : Exécution de la suppression d'une voiture

7.2 La gestion des clients

- Remplissage

Ci-joint la figure 33 montre le code de remplissage du tableau des clients.

```
case 2:
    printf("Vous avez choisi d'ajouter des clients.\n");
    printf("\n\nCombien de client voulez vous ajouter ? ");
    scanf("%d", &nbClients);
    clientsAgence=allocationclient(nbClients);
    printf("\n*****Ajout des clients*****\n");
    remplirFichierClientsIndex (fp, fi, stockvoiture, nbVoitures, clientsAgence, nbClients);
    break;
break;
```

Fig. 30 : Code de remplissage du tableau clients

```
*****Ajout des clients*****
Saisir le CIN du client 14010163
Saisir le nom du client Jomaa
Saisir le prenom du client MedRayen
Saisir le telephone du client 97332857
Saisir le nombre de locations de ce client 2

LOCATION 1
Donner le code de voiture a louer pour le client Jomaa MedRayen 1
```

Fig. 31 : Exécution du remplissage du tableau clients

- **Affichage**

Ci-joint la figure 33 montre le code d'affichage du tableau des clients.

```
case 1:
    printf("\n*****Affichage*****\n");
    afficherFichierClientsIndex(fp, fi, nbClients);
break;
```

Fig. 32 : Code d'affichage du tableau clients

```
*****Affichage*****

Affichage Fichier index
Le CIN du client: 14010163
Le nom et prnom du client: Jomaa Rayen
Le telephone du client : 97332857
Le nombre de ces locations: 0
Codes locations

Le CIN du client: 4189808
Le nom et prnom du client: Bouhachem Amine
Le telephone du client : 198
Le nombre de ces locations: 0
Codes locations
```

Fig. 33 : Excution de l'affichage du tableau clients

- **Le client qui a le nombre de locations le plus élevé**

Ci-joint la figure 35 montre le code d’affichage du tableau des clients.

```
case 4:
    printf("Vous avez choisi de trouver le client avec le plus de locations.\n");
    if (tableauResultats == NULL) {
        printf("Erreur lors de l'allocation du tableau de résultats\n");
        exit(-1);
    }
    //remplissage du tableau dynamique d'adresses
    remplirTabResultat(tableauResultats, nbResultats, clientsAgence, nbClients, capaciteResultats);
    // Trouver le client avec le montant le plus élevé à payer
    maxNbLocations(tableauResultats, nbClients);
    // Libérer la mémoire allouée pour chaque résultat
    for (i = 0; i < nbResultats; i++) {
        free(tableauResultats[i]);
    }
break;
```

```
void maxNbLocations(RESULTAT** tableauResultats, int nbClients){
    int indexMax = 0;
    int nbMax = (*tableauResultats+0)->nbLocations;
    printf("\n-----\n");
    int i;
    for (i = 1; i < nbClients; i++) {
        if ((*tableauResultats+i)->nbLocations > nbMax) {
            nbMax = (*tableauResultats+i)->nbLocations;
            indexMax = i;
        }
    }

    // Afficher les détails du client ayant le montant le plus élevé à payer
    printf("Client avec le nombre de location le plus élevé à payer :\n");
    printf("Nom et prénom du client : %s %s\n", (*tableauResultats+indexMax)->nomClient,
        tableauResultats[indexMax]->prenomClient);
    printf("Nombre de location : %d\n", (*tableauResultats+indexMax)->nbLocations);
}
```

Fig. 34 : Code d’affichage du client qui a le plus grand nombre de locations

```
***** MENU CLIENTS *****

1 - Afficher les clients (a partir du fichier)
2 - Ajouter des clients
3 - Modifier un client
4 - Trouver le client avec le plus de locations
5 - Afficher fichier index
6 - Revenir au menu principal
Choisissez une option : 4
Vous avez choisi de trouver le client avec le plus de locations.

-----
Client avec le nombre de location le plus élevé à payer :
Nom et prénom du client : 
Nombre de location :
```

Fig. 35 : Exécution d'affichage du client qui a le plus grand nombre de locations

Conclusion générale

Conclusion générale

En conclusion, la création d'une application de gestion pour une agence de location de voitures en langage C a été une expérience enrichissante. Ce projet m'a offert l'opportunité d'approfondir mes connaissances en programmation et de développer des compétences précieuses dans plusieurs domaines.

Travailler en langage C m'a permis de maîtriser les aspects fondamentaux de la programmation, notamment la gestion de la mémoire, la manipulation des pointeurs et la modularité du code. La conception modulaire du système, avec ses différents modules pour la gestion des véhicules, des clients, et des locations, a renforcé ma compréhension de l'organisation logicielle.

Annexes