RAPPORT :

# Mini projet : programmation en C

# Sujet :Location de voiture



**Objectifs :** nous avons pour objectif de développer une **application console** pour la gestion de la location de voitures , en stockant les informations des voitures et des utilisateurs dans une base de données « fichier .csv », et en créant une interface utilisateur conviviale pour faciliter l’utilisation de l’application.

réalisé par :

* Imrane lakniti
* Mehdi

derbez

# Fichier header

max\_char(Voiture t, int n, int l) : Cette fonction calcule la longueur maximale de chaque champ de texte dans les structures Voiture. Elle prend en entrée un tableau de structures Voiture, la taille de ce tableau et un tableau l pour stocker les longueurs maximales. Elle parcourt chaque voiture dans le tableau et compare la longueur de ses champs de texte avec les longueurs maximales actuellement stockées dans le tableau l. Si une longueur est plus grande que celle stockée, elle met à jour la valeur dans l.

taille\_int(int n) : Cette fonction calcule la taille d'un entier (le nombre de chiffres qu'il contient). Elle prend en entrée un entier n et retourne le nombre de chiffres dans n en divisant répétitivement n par 10 jusqu'à ce que n devienne 0, en comptant le nombre d'itérations nécessaires.

max\_int(Voiture t, int n, int l) : Cette fonction est similaire à max\_char, mais elle calcule la longueur maximale des entiers dans les structures Voiture.

printf\_Tirer(int n) : Cette fonction affiche un trait d'underscore (\_) répété n fois.

printf\_Space(int n) : Cette fonction affiche un espace ( ) répété n fois.

Down(int ti, int tc, int n) : Cette fonction affiche une ligne séparatrice dans le tableau, avec des traits d'underscore séparant chaque colonne, dont la longueur est déterminée par les tableaux ti (pour les entiers) et tc (pour les chaînes de caractères) passés en paramètres.

head(Voiture t, int n) : Cette fonction affiche l'en-tête du tableau, décrivant les différents champs des voitures. Elle appelle la fonction Down pour afficher la ligne de séparation après avoir affiché l'en-tête.

aff(Voiture t, int i, int n) : Cette fonction affiche les détails d'une voiture spécifique, identifiée par l'indice i, dans le tableau. Elle utilise les mêmes méthodes que head pour aligner correctement les données.

**#include <stdio.h> :** Ce fichier d'en-tête fournit des fonctions pour l'entrée et la sortie standard. Il contient les définitions des fonctions comme printf() pour afficher des données à l'écran, scanf() pour lire les données à partir de l'entrée standard (clavier) et des fonctions pour manipuler les fichiers, telles que fopen(), fclose(), fprintf(), fscanf() pour travailler avec des fichiers.

#include <string.h> : Ce fichier d'en-tête fournit des fonctions pour manipuler les chaînes de caractères en langage C. Il contient des fonctions comme strcpy() pour copier une chaîne dans une autre, strcmp() pour comparer deux chaînes, strcat() pour concaténer des chaînes, etc.

#include <stdlib.h> : Ce fichier d'en-tête fournit des fonctions pour la gestion de la mémoire dynamique et d'autres fonctions utilitaires. Il contient des fonctions telles que malloc(), calloc(), realloc() pour allouer et libérer la mémoire, atoi() pour convertir une chaîne en entier, atof() pour convertir une chaîne en nombre à virgule flottante, etc.

#include <unistd.h> : Ce fichier d'en-tête est spécifique aux systèmes de type Unix et fournit un accès à certaines fonctionnalités du système d'exploitation Unix, telles que la fonction sleep(), qui suspend l'exécution du programme pendant un certain nombre de secondes.

#include "Tableau.h" : Cela inclut le fichier d'en-tête "Tableau.h" qui est spécifique à mon programme. Ce fichier contient des déclarations de fonctions, des définitions de structures de données ou d'autres macros utilisées dans le programme principal. En incluant ce fichier, vous pouvez réutiliser ces déclarations et définitions dans votre programme principal sans avoir à les recopier à chaque fois.

Initialisation du programme :

Le programme commence par inclure les fichiers d'en-tête nécessaires, tels que <stdio.h> , <string.h>, <stdlib.h>, <unistd.h> et "Tableau.h".

Ensuite, le programme déclare quelques variables globales, telles que Nom pour stocker le nom de l'utilisateur et ID\_TXT pour stocker le dernier ID utilisé.

La fonction Animation() est appelée, affichant une animation de bienvenue et demandant à l'utilisateur de saisir son nom.

Lecture des données existantes (s'il y en a) :

La fonction countLines() est utilisée pour compter le nombre de lignes dans le fichier "Data.csv", qui contient les données des voitures.

Si des données existent, la mémoire est allouée pour stocker ces données dans un tableau de structures de voitures, et la fonction Read\_Data() est appelée pour lire les données à partir du fichier CSV et les stocker dans le tableau.

Menu principal :

Le programme entre ensuite dans une boucle qui affiche le menu principal à l'utilisateur, lui permettant de choisir parmi différentes options, telles que l'affichage des voitures, l'ajout, la suppression, la modification, la recherche ou le tri des voitures, ou la sortie du programme.

Selon le choix de l'utilisateur, le programme appelle la fonction appropriée pour effectuer l'action demandée.

Fonctionnalités principales :

Les fonctions telles que afficher(), Ajout(), modifier(), supression(), Recherche\_Marque(), Recherche\_Dis(), Trier\_Marque() et Trier\_Prix() implémentent les fonctionnalités principales du programme, telles que l'affichage, l'ajout, la suppression, la modification, la recherche et le tri des voitures.

Écriture des données :

Après chaque modification des données, le programme utilise la fonction write\_Data() pour écrire les données mises à jour dans le fichier CSV "Data.csv".

Nettoyage et sortie :

À la fin de chaque itération de la boucle du menu principal, le programme attend que l'utilisateur appuie sur une touche pour continuer, puis efface l'écran et affiche à nouveau le menu principal.

Libération de la mémoire :

Lorsque l'utilisateur choisit de quitter le programme, la mémoire allouée pour le tableau de voitures est libérée à l'aide de la fonction free().

Déclarations globales :

Nom est un tableau de caractères utilisé pour stocker un nom.

ID\_TXT est une variable globale utilisée pour stocker un identifiant.

Fonction WriteID() :

Cette fonction écrit la valeur actuelle de ID\_TXT dans un fichier nommé "Id.txt".

Si le fichier peut être ouvert, ID\_TXT est incrémenté et écrit dans le fichier.

Fonction ReadID() :

Cette fonction lit la valeur stockée dans le fichier "Id.txt" et la stocke dans ID\_TXT.

Fonction Inserer() :

La fonction Inserer() est responsable de l'insertion des données d'une nouvelle voiture dans un tableau de structures de voitures .

1. **Saisie des données de la nouvelle voiture :** La fonction demande à l'utilisateur de saisir les différentes informations concernant la nouvelle voiture, telles que la marque, le modèle, le type de carburant, le nombre de places, le type de transmission, le prix et la disponibilité.
2. **Traitement des données de carburant et de transmission :** La fonction demande à l'utilisateur de saisir un code pour le type de carburant et le type de transmission, puis utilise des conditions pour assigner les valeurs correspondantes à ces attributs de la voiture (**t[i].type\_C** et **t[i].type\_T**).
3. **Génération de l'identifiant :** La fonction lit l'identifiant actuel à partir d'un fichier texte, incrémente cette valeur, écrit la nouvelle valeur dans le fichier texte, puis assigne cette nouvelle valeur à l'attribut **Id** de la nouvelle voiture (**t[i].Id**).
4. **Assignation du nom de l'utilisateur :** La fonction copie le nom de l'utilisateur, stocké dans la variable globale **Nom**, dans l'attribut **nom** de la nouvelle voiture (**t[i].nom**).

Fonction countLines() :

1. **Ouverture du fichier :** La fonction tente d'ouvrir le fichier spécifié, "Data.csv", en mode lecture.
2. **Comptage des lignes :** Une fois le fichier ouvert avec succès, la fonction commence à parcourir le fichier caractère par caractère en utilisant **fgetc()**. À chaque caractère lu, elle vérifie s'il s'agit d'un retour à la ligne (**\n**). Chaque fois qu'un retour à la ligne est rencontré, cela signifie qu'une ligne est terminée, donc elle incrémente un compteur de lignes.
3. **Fermeture du fichier :** Une fois que toutes les lignes du fichier ont été comptées, la fonction ferme le fichier.
4. **Retour du nombre de lignes :** La fonction retourne le nombre total de lignes comptées dans le fichier.

Fonction Ajout() :

La fonction **Ajout()** est chargée d'ajouter une nouvelle voiture au tableau existant. Voici une explication succincte de son fonctionnement :

1. **Incrémentation de la taille du tableau** : La fonction incrémente d'abord la taille du tableau en incrémentant la valeur pointée par **n**, qui représente le nombre total de voitures dans le tableau.
2. **Allocation de mémoire** : Si le tableau est encore vide (**NULL**), la fonction alloue de la mémoire pour le premier élément du tableau en utilisant **malloc()**. Sinon, elle redimensionne le tableau existant à l'aide de **realloc()**.
3. **Insertion des données de la nouvelle voiture** : Une fois la mémoire allouée, la fonction appelle **Inserer()** pour saisir les données de la nouvelle voiture dans le tableau, en passant l'index correct pour l'insertion.
4. **Retour du tableau mis à jour** : Enfin, la fonction retourne le tableau mis à jour, prêt à être utilisé avec la nouvelle voiture ajoutée.

Fonction Menu\_M() :

Cette fonction affiche un menu pour la modification des données d'une voiture et retourne le choix de l'utilisateur.

Fonction modifier():

1. Saisie de l'identifiant de la voiture à modifier : L'utilisateur est invité à saisir l'identifiant de la voiture qu'il souhaite modifier.
2. Recherche de la voiture : La fonction parcourt le tableau pour rechercher une voiture ayant l'identifiant saisi. Si une correspondance est trouvée, la fonction passe à l'étape suivante. Sinon, elle informe l'utilisateur qu'aucune voiture correspondante n'a été trouvée et arrête la modification.
3. Affichage du menu de modification : Un menu est affiché à l'utilisateur, lui permettant de choisir quel attribut de la voiture il souhaite modifier : la marque, le nom, le modèle, le type de carburant, le nombre de places, le type de transmission, le prix ou la disponibilité.
4. Modification de l'attribut sélectionné : Selon le choix de l'utilisateur, la fonction lui demande de saisir la nouvelle valeur pour l'attribut sélectionné de la voiture trouvée.
5. Mise à jour du tableau : Une fois que toutes les modifications ont été effectuées, le tableau est mis à jour avec les nouvelles valeurs de la voiture modifiée.

Fonction supression() :

* Recherche de la voiture à supprimer : La fonction parcourt le tableau pour rechercher une voiture ayant l'identifiant saisi par l'utilisateur. Si une correspondance est trouvée, la fonction supprime cette voiture du tableau en décalant les éléments suivants pour remplir l'espace laissé par la voiture supprimée. Si aucune correspondance n'est trouvée, un message informe l'utilisateur qu'aucune voiture correspondante n'a été trouvée.
* Mise à jour de la taille du tableau : Après la suppression de la voiture, la fonction réduit la taille du tableau en utilisant realloc() pour libérer la mémoire allouée précédemment pour la voiture supprimée. Ensuite, elle retourne le tableau mis à jour.

Fonction afficher() :

Cette fonction affiche les données de toutes les voitures dans le tableau.

La fonction Read\_Data() :

1. Ouverture du fichier : La fonction tente d'ouvrir le fichier "Data.csv" en mode lecture à l'aide de fopen().
2. Lecture des données : Une fois le fichier ouvert avec succès, la fonction commence à lire son contenu ligne par ligne à l'aide de fgets(). Pour chaque ligne lue, elle utilise strtok() pour diviser la ligne en tokens (chaînes de caractères), en utilisant la virgule comme délimiteur, afin d'extraire les différentes valeurs des attributs de la voiture (comme l'identifiant, la marque, le modèle, etc.).
3. Stockage des données dans le tableau : Les valeurs extraites sont converties en types appropriés (comme int pour l'identifiant et float pour le prix) et stockées dans les attributs correspondants de la structure Voiture dans le tableau.
4. Fermeture du fichier : Une fois toutes les données lues et stockées, le fichier est fermé à l'aide de fclose().

fonction write\_Data() :

Ouverture du fichier : La fonction tente d'ouvrir le fichier "Data.csv" en mode écriture ("wt") à l'aide de fopen().

Écriture des données : Une fois le fichier ouvert avec succès, la fonction boucle à travers chaque voiture dans le tableau. Pour chaque voiture, elle utilise fprintf() pour écrire les valeurs de ses attributs dans le fichier, en les séparant par des virgules et en respectant l'ordre des colonnes défini dans le fichier CSV.

Fermeture du fichier : Une fois toutes les données écrites dans le fichier, la fonction le ferme en utilisant fclose().

Recherche par Marque (Recherche\_Marque())

Saisie de la Marque : L'utilisateur est invité à saisir la marque de voiture qu'il souhaite rechercher.

Parcours du Tableau : La fonction parcourt le tableau de voitures. Pour chaque voiture, elle compare la marque saisie par l'utilisateur avec la marque de la voiture dans le tableau.

Affichage des Résultats : Si une correspondance est trouvée, la voiture est affichée. Ce processus continue jusqu'à ce que toutes les voitures du tableau aient été vérifiées.

Gestion des Erreurs : Si aucune correspondance n'est trouvée, la fonction affiche un message informant l'utilisateur qu'aucune voiture correspondante n'a été trouvée.

Recherche par Disponibilité (Recherche\_Dis())

Saisie de la Disponibilité : L'utilisateur est invité à saisir le statut de disponibilité qu'il souhaite rechercher (par exemple, disponible ou non disponible).

Parcours du Tableau : La fonction parcourt le tableau de voitures. Pour chaque voiture, elle compare le statut de disponibilité saisi par l'utilisateur avec le statut de disponibilité de la voiture dans le tableau.

Affichage des Résultats : Si une correspondance est trouvée, la voiture est affichée. Ce processus continue jusqu'à ce que toutes les voitures du tableau aient été vérifiées.

Gestion des Erreurs : Si aucune correspondance n'est trouvée, la fonction affiche un message informant l'utilisateur qu'aucune voiture correspondante n'a été trouvée.

Tri par Marque (Trier\_Marque()) :

Parcours du Tableau : La fonction parcourt le tableau de voitures et compare les marques de chaque paire de voitures consécutives.

Échange de Positions : Si la marque de la voiture actuelle est supérieure à la marque de la voiture suivante (en ordre alphabétique), les voitures sont échangées de position dans le tableau.

Répétition du Processus : Ce processus est répété jusqu'à ce que toutes les voitures du tableau aient été comparées et triées en fonction de leur marque.

Résultat : À la fin du tri, les voitures sont ordonnées par ordre alphabétique de leur marque dans le tableau.

Tri par Prix (Trier\_Prix()) :

Parcours du Tableau : La fonction parcourt le tableau de voitures et compare les prix de chaque paire de voitures consécutives.

Échange de Positions : Si le prix de la voiture actuelle est supérieur au prix de la voiture suivante, les voitures sont échangées de position dans le tableau.

Répétition du Processus : Ce processus est répété jusqu'à ce que toutes les voitures du tableau aient été comparées et triées en fonction de leur prix.

Résultat : À la fin du tri, les voitures sont ordonnées par ordre croissant de leur prix dans le tableau.

\*/