Rapport de projet de location de voitures :

Introduction :

Le code en langage C que nous présentons ici est un système de location de voitures, conçu pour faciliter la gestion et la réservation de véhicules dans une entreprise de location. Ce projet a été développé dans le but d'automatiser le processus de location, d'améliorer l'efficacité opérationnelle et de fournir une interface conviviale pour les utilisateurs.

Ce projet met en œuvre des fonctionnalités clés telles que la création de profils clients, la gestion des réservations, la vérification de la disponibilité des véhicules, le calcul des tarifs de location, et la génération de rapports. Il offre également une interface utilisateur intuitive qui facilite la navigation et l'interaction avec le système.

Ce rapport présente une vue d'ensemble détaillée du code de location de voitures en langage C, en expliquant les fonctionnalités clés, la structure du programme, ainsi que les choix de conception et d'implémentation effectués. Il vise à fournir une compréhension approfondie du système et à servir de guide pour les développeurs souhaitant étendre ou personnaliser le code.

Explication des structures :

1. Structure Car :

• id : Un entier représentant l'identifiant de la voiture.

• brand : Un tableau de caractères représentant la marque de la voiture.

• model : Un tableau de caractères représentant le modèle de la voiture.

• fuelType : Un tableau de caractères représentant le type de carburant de la voiture.

• numberOfSeats : Un entier représentant le nombre de sièges dans la voiture.

• transmission : Un tableau de caractères représentant le type de transmission de la voiture.

• rentalPricePerDay : Une valeur flottante représentant le prix de location par jour de la voiture.

• availability : Un entier indiquant la disponibilité de la voiture (1 pour disponible, 0 pour non disponible).

• age : Un entier représentant l'âge de la voiture.

2. Structure Client :

• clientid : Un entier représentant l'identifiant du client.

• username : Un tableau de caractères représentant le nom d'utilisateur du client.

• address : Un tableau de caractères représentant l'adresse du client.

3. Structure Rental:

• ID : Un entier représentant l'identifiant de la location.

• clientId : Un entier représentant l'identifiant du client associé à la location.

• car : Une instance de la structure Car représentant la voiture louée.

• startDate : Un tableau de caractères représentant la date de début de la location.

• endDate : Un tableau de caractères représentant la date de fin de la location.

• dailyRate : Une valeur flottante représentant le tarif journalier de location.

• additionalFees : Une valeur flottante représentant les frais supplémentaires associés à la location.

4. Structure Employee :

• id1 : Un entier représentant l'identifiant de l'employé.

• name : Un tableau de caractères représentant le nom de l'employé.

• position : Un tableau de caractères représentant la fonction de l'employé.

5. Structure Reservation :

• id2 : Un entier représentant l'identifiant de la réservation.

• carId : Un entier représentant l'identifiant de la voiture associée à la réservation.

• clientId : Un entier représentant l'identifiant du client associé à la réservation.

• startDate : Un tableau de caractères représentant la date de début de la réservation.

• endDate : Un tableau de caractères représentant la date de fin de la réservation.

• totalCost : Une valeur flottante représentant le coût total de la réservation.

De plus, le code inclut plusieurs directives de préprocesseur et définitions de constantes :

• MAX\_CAR\_SIZE, MAX\_BRAND\_SIZE, MAX\_MODEL\_SIZE, MAX\_TRANSMISSION\_SIZE, MAX\_FUELTYPE\_SIZE, MAX\_CLIENT\_SIZE, MAX\_EMPLOYEE\_SIZE, MAX\_STRING\_LENGTH : Ces constantes définissent la taille maximale (en caractères) des différents champs de chaînes de caractères utilisés dans les structures.

• Le code inclut également les bibliothèques standard du langage C telles que stdio.h, stdlib.h, string.h et time.h pour les opérations d'entrée/sortie, la gestion de la mémoire, la manipulation de chaînes de caractères et les fonctions liées au temps, respectivement.

Explication des fonctions :

Le code contient plusieurs fonctions liées à la gestion d'un système de location de voitures. Voici une explication de chaque fonction :

welcomeMessage() : Cette fonction affiche un message de bienvenue dans le système de gestion de location de voitures. Le message affiché est "Welcome to the Car Rental Management System".

clearBuffer() : Cette fonction est utilisée pour vider le tampon d'entrée. Elle lit les caractères du tampon jusqu'à ce qu'elle atteigne un retour à la ligne ('\n') ou la fin du fichier (EOF). Cela permet de nettoyer le tampon après une saisie d'utilisateur.

printCar(Car car) : Cette fonction prend en paramètre une structure de type Car et affiche les détails de la voiture. Les informations affichées incluent l'ID, la marque, le modèle, le type de carburant, le nombre de sièges, la transmission, le prix de location par jour, la disponibilité et l'âge de la voiture.

addCar() : Cette fonction permet d'ajouter une nouvelle voiture au système. Elle ouvre le fichier "cars.csv" en mode ajout ("a") et récupère les détails de la nouvelle voiture à partir de l'utilisateur. Les détails comprennent l'ID, la marque, le modèle, le type de carburant, le nombre de sièges, la transmission, le prix de location par jour, la disponibilité et l'âge de la voiture. Les saisies de l'utilisateur sont stockées dans une structure Car appelée newCar. Les informations de newCar sont ensuite écrites dans le fichier CSV à l'aide de la fonction fprintf(). Enfin, le fichier est fermé et un message de confirmation est affiché.

addClient() : Cette fonction permet d'ajouter un nouveau client au système de gestion de location de voitures. Elle ouvre le fichier "clients.csv" en mode ajout ("a") et récupère les détails du nouveau client à partir de l'utilisateur. Les détails comprennent l'ID du client, le nom du client et l'adresse du client. Les saisies de l'utilisateur sont stockées dans une structure Client appelée client. Les informations de client sont ensuite écrites dans le fichier CSV à l'aide de la fonction fprintf(). Enfin, le fichier est fermé et un message de confirmation est affiché.

modifyCar() : Cette fonction permet de modifier les détails d'une voiture existante dans le système. Elle demande à l'utilisateur d'entrer l'ID de la voiture à modifier, puis charge toutes les voitures à partir du fichier "cars.csv" dans un tableau de structures Car. La fonction recherche la voiture avec l'ID correspondant et enregistre son index dans la variable index. Si la voiture n'est pas trouvée, un message d'erreur est affiché. Si la voiture est trouvée, la fonction demande à l'utilisateur d'entrer les nouvelles informations pour la voiture, telles que la marque, le modèle, le type de carburant, le nombre de sièges, la transmission, le prix de location par jour, la disponibilité et l'âge. Les nouvelles informations sont stockées dans la structure car[index]. Ensuite, le fichier "cars.csv" est réécrit avec les données mises à jour. Le fichier est ouvert en mode écriture ("w") et chaque voiture du tableau car est écrite dans le fichier à l'aide de la fonction fprintf(). Enfin, le fichier est fermé et un message de confirmation est affiché.

deleteCar() : Cette fonction permet de supprimer une voiture du système en fonction de son ID. Elle demande à l'utilisateur d'entrer l'ID de la voiture à supprimer, puis charge toutes les voitures à partir du fichier "cars.csv" dans un tableau de structures Car. La fonction recherche la voiture avec l'ID correspondant et enregistre son index dans la variable index. Si la voiture n'est pas trouvée, un message d'erreur est affiché. Si la voiture est trouvée, la fonction réécrit le fichier "cars.csv" en excluant la voiture à supprimer. Le fichier est ouvert en mode écriture ("w") et chaque voiture du tableau cars est écrite dans le fichier à l'aide de la fonction fprintf(). Enfin, le fichier est fermé et un message de confirmation est affiché.

addRental() : Cette fonction permet d'ajouter une nouvelle location de voiture au système. Elle ouvre le fichier "rentals.csv" en mode ajout ("a") et récupère les détails de la nouvelle location à partir de l'utilisateur. Les détails comprennent le nom d'utilisateur du client, l'ID de la voiture, la date de début, la date de fin, le tarif journalier et les frais supplémentaires. Les saisies de l'utilisateur sont stockées dans une structure Rental appelée newRental. Les informations de newRental sont ensuite écrites dans le fichier CSV à l'aide de la fonction fprintf(). Enfin, le fichier est fermé et un message de confirmation est affiché.

updateClient() : Cette fonction permet de mettre à jour les informations d'un client existant dans le système. Elle demande à l'utilisateur d'entrer l'ID du client à mettre à jour, puis ouvre le fichier "clients.csv" en mode lecture et écriture ("r+"). La fonction parcourt le fichier ligne par ligne et recherche le client avec l'ID correspondant. Si le client est trouvé, la fonction demande à l'utilisateur d'entrer les nouvelles informations pour le client, telles que le nom et l'adresse. Les nouvelles informations sont stockées dans la structure client. Ensuite, la fonction utilise la fonction fseek() pour déplacer le pointeur du fichier vers le début de la ligne correspondante, puis utilise la fonction fprintf() pour réécrire la ligne avec les nouvelles informations. Enfin, le fichier est fermé et un message de confirmation est affiché. Si aucun client n'est trouvé avec l'ID spécifié, un message d'erreur est affiché.

generateInvoice() : Cette fonction génère une facture pour une location de voiture. Elle commence par ouvrir le fichier "cars.csv" en mode ajout ("a"). Ensuite, elle utilise la fonction fprintf() pour écrire les détails de la facture dans le fichier. Les détails incluent le client, la voiture, la durée de location et le coût total. Enfin, les fichiers sont fermés et un message de confirmation est affiché.

searchCars() : Cette fonction permet de rechercher des voitures en fonction de la marque et de la disponibilité. Elle demande à l'utilisateur d'entrer la marque de la voiture à rechercher et la disponibilité souhaitée. Ensuite, elle ouvre le fichier "cars.csv" en mode lecture ("r") et le fichier "search\_results.csv" en mode écriture ("w"). La fonction parcourt chaque ligne du fichier et vérifie si la marque et la disponibilité correspondent aux critères de recherche. Si une correspondance est trouvée, la voiture est écrite dans le fichier "search\_results.csv" à l'aide de la fonction fprintf(). Si aucune voiture n'est trouvée, un message approprié est écrit dans le fichier. Enfin, les fichiers sont fermés et un message de confirmation est affiché.

CarReservation() : Cette fonction permet de rechercher et de réserver des voitures disponibles. Elle ouvre le fichier "cars.csv" en mode lecture ("r") et le fichier "reservation\_results.csv" en mode écriture ("w"). La fonction parcourt chaque ligne du fichier et vérifie si la voiture est disponible. Si une voiture disponible est trouvée, elle est ajoutée à un tableau cars de taille maximale MAX\_CAR\_SIZE. Une fois toutes les voitures disponibles trouvées, la fonction écrit les réservations dans le fichier "reservation\_results.csv" à l'aide de la fonction fprintf(). Si aucune voiture n'est disponible, un message approprié est écrit dans le fichier. Enfin, les fichiers sont fermés et un message de confirmation est affiché.

checkReservationAvailability() : Cette fonction permet de vérifier la disponibilité d'une réservation de voiture. Elle demande à l'utilisateur d'entrer l'identifiant de la voiture, la date de début et la date de fin de la réservation. Ensuite, elle ouvre le fichier "cars.csv" en mode lecture ("r"). La fonction lit chaque ligne du fichier et extrait les informations de réservation. Si une réservation correspond à l'identifiant de la voiture et chevauche les dates de réservation spécifiées, un message indiquant que la voiture n'est pas disponible est affiché. Sinon, un message indiquant que la voiture est disponible est affiché. Enfin, le fichier est fermé.

cancelReservation() : Cette fonction permet d'annuler des réservations de voiture. Elle ouvre le fichier "cars.csv" en mode lecture ("r") et le fichier "cancel\_results.csv" en mode écriture ("w"). La fonction lit chaque ligne du fichier et vérifie si la voiture est réservée et non disponible. Si une réservation correspond à ces critères, la voiture est ajoutée à un tableau voitures de taille maximale MAX\_CAR\_SIZE. Une fois toutes les voitures réservées trouvées, la fonction écrit les résultats dans le fichier "cancel\_results.csv" en utilisant la fonction fprintf(). Si aucune réservation n'est trouvée, un message approprié est écrit dans le fichier. Enfin, les fichiers sont fermés et un message de confirmation est affiché.

compareByBrand(), compareByPrice() et compareByAvailability() : Ce sont des fonctions de comparaison utilisées par la fonction sortCars() pour trier les voitures selon la marque, le prix de location par jour et la disponibilité. Ces fonctions sont utilisées par la fonction qsort() de la bibliothèque standard pour effectuer le tri.

sortCars() : Cette fonction permet de trier les voitures en fonction d'une option choisie par l'utilisateur. Elle ouvre le fichier "cars.csv" en mode lecture ("r") et compte le nombre total de voitures dans le fichier. Ensuite, elle demande à l'utilisateur de choisir l'option de tri (par marque, prix de location par jour ou disponibilité). En fonction de l'option choisie, la fonction utilise la fonction qsort() pour trier les voitures. Enfin, elle affiche les voitures triées en utilisant la fonction printCar(). Le fichier est ensuite fermé.

recommendCarModel() : Cette fonction recommande des modèles de voitures en fonction du nombre de sièges requis. Elle demande à l'utilisateur d'entrer le nombre de sièges nécessaires, puis ouvre le fichier "cars.csv" en mode lecture ("r"). La fonction lit chaque ligne du fichier, extrait les informations de la voiture et vérifie si la voiture est disponible et a un nombre de sièges supérieur ou égal au nombre de sièges requis. Si ces conditions sont remplies, la fonction affiche les détails de la voiture en utilisant la fonction printCar(). Si aucune voiture n'est trouvée, un message approprié est affiché. Enfin, le fichier est fermé.

sellCarsOver5Years() : Cette fonction affiche les voitures de plus de 5 ans disponibles à la vente. Elle ouvre le fichier "cars.csv" en mode lecture ("r") et ignore la première ligne (les en-têtes). Ensuite, elle lit chaque ligne du fichier, extrait les informations de la voiture et vérifie si la voiture a plus de 5 ans et est disponible. Si ces conditions sont remplies, la fonction affiche les détails de la voiture. Enfin, le fichier est fermé.

calculateRentalCost() : Cette fonction permet de calculer le coût total de location d'une voiture pour un nombre de jours spécifié. Elle demande à l'utilisateur d'entrer l'identifiant de la voiture et le nombre de jours de location. Ensuite, elle ouvre le fichier "rental.csv" en mode lecture ("r"). La fonction lit chaque ligne du fichier, extrait l'identifiant et le prix de location de la voiture, et vérifie si l'identifiant correspond à celui saisi par l'utilisateur. Si une correspondance est trouvée, la fonction calcule le coût total de location en multipliant le nombre de jours par le prix de location par jour, puis affiche le coût total. Si aucune correspondance n'est trouvée, un message indiquant que l'identifiant de la voiture n'a pas été trouvé est affiché. Enfin, le fichier est fermé.

addEmployee() : Cette fonction permet d'ajouter un employé au fichier "employee.csv". Elle demande à l'utilisateur d'entrer l'identifiant de l'employé, le nom et le poste. Ensuite, elle écrit les informations de l'employé dans le fichier en utilisant la fonction fprintf(). Enfin, un message de confirmation est affiché.

modifyEmployee() : Cette fonction permet de modifier les informations d'un employé dans le fichier "employee.csv". Elle demande à l'utilisateur d'entrer l'identifiant de l'employé à modifier. Ensuite, elle ouvre le fichier "employee.csv" en mode lecture ("r") et le fichier temporaire "temp.csv" en mode écriture ("w"). La fonction lit chaque ligne du fichier, extrait les informations de l'employé et vérifie si l'identifiant correspond à celui saisi par l'utilisateur. Si une correspondance est trouvée, la fonction demande à l'utilisateur d'entrer le nouveau nom et le nouveau poste de l'employé, puis les écrit dans le fichier temporaire. Si aucune correspondance n'est trouvée, la ligne est simplement copiée du fichier d'origine au fichier temporaire. Enfin, les fichiers sont fermés, le fichier d'origine est supprimé et le fichier temporaire est renommé en "employee.csv".

deleteEmployee() : Cette fonction permet de supprimer un employé du fichier "employee.csv". Elle demande à l'utilisateur d'entrer l'identifiant de l'employé à supprimer. Ensuite, elle ouvre le fichier "employee.csv" en mode lecture ("r") et le fichier temporaire "temp.csv" en mode écriture ("w"). La fonction lit chaque ligne du fichier, extrait les informations de l'employé et vérifie si l'identifiant correspond à celui saisi par l'utilisateur. Si une correspondance est trouvée, la ligne est simplement ignorée. Si aucune correspondance n'est trouvée, la ligne est copiée du fichier d'origine au fichier temporaire. Enfin, les fichiers sont fermés, le fichier d'origine est supprimé et le fichier temporaire est renommé en "employee.csv".

manageEmployees() : est une fonction qui gère les opérations liées aux employés. Voici une explication étape par étape de son fonctionnement :

La fonction commence par ouvrir le fichier "employee.csv" en mode ajout et lecture ("a+") à l'aide de la fonction fopen(). Si le fichier ne peut pas être ouvert, c'est-à-dire si file est égal à NULL, un message d'erreur est affiché indiquant que le fichier n'a pas pu être ouvert. Ensuite, la fonction se termine avec return pour sortir de la fonction.Si le fichier est ouvert avec succès, la fonction présente un menu à l'utilisateur en utilisant printf(). Le menu propose trois options : ajouter un employé, modifier un employé ou supprimer un employé. L'utilisateur est invité à entrer son choix en utilisant scanf(), et la valeur est stockée dans la variable choice. La fonction utilise ensuite clearBuffer() pour vider le tampon d'entrée, ce qui permet d'éliminer les caractères de nouvelle ligne résiduels laissés par l'appel précédent à scanf(). Ensuite, la fonction utilise une instruction switch pour exécuter le code correspondant à l'option choisie par l'utilisateur. Si le choix est 1, la fonction appelle addEmployee(file) pour ajouter un employé. Si le choix est 2, la fonction appelle modifyEmployee(file) pour modifier un employé. Si le choix est 3, la fonction appelle deleteEmployee(file) pour supprimer un employé. Si le choix ne correspond à aucune des options, la fonction affiche un message d'erreur indiquant que le choix est invalide. Une fois que l'opération correspondante a été effectuée, la fonction utilise fclose(file) pour fermer le fichier "employee.csv" .

Explications du main :

La fonction main() définit un certain nombre de prototypes de fonctions pour des fonctions qui seront appelées plus tard dans le programme. Ces prototypes de fonctions indiquent au compilateur que ces fonctions seront définies ailleurs dans le code.

Ensuite, la fonction main() utilise une boucle do-while pour afficher un menu principal à l'utilisateur et lui permettre de choisir différentes options. Le menu affiche différentes fonctionnalités du système de gestion de location de voitures, telles que l'ajout d'une nouvelle voiture, la modification d'une voiture, l'ajout d'un nouveau client, etc.

L'utilisateur est invité à entrer son choix en utilisant scanf(), et la valeur est stockée dans la variable choice.

La fonction utilise ensuite une instruction switch pour exécuter le code correspondant à l'option choisie par l'utilisateur. Chaque cas correspond à une option du menu et appelle la fonction correspondante.

Si l'utilisateur choisit l'option 0, le programme affiche un message de sortie et la boucle do-while se termine, ce qui entraîne la fin du programme.

Si l'utilisateur choisit une option invalide, un message d'erreur est affiché.

La boucle do-while continue tant que l'utilisateur ne choisit pas l'option 0 pour quitter le programme.

Une fois la boucle terminée, la fonction main() se termine et le programme se termine.