RAPPORT

GOSH VERY GOOD TRIP !

« BLABLACAR DU WISH »

DADOUN Ambre
WITHANAGE Perera Sakun
LABBACI Amira

A- Organisation du programme

Le programme utilise diverses fonctions pour plus de lisibilité, clarifier le code et éviter les répétitions de code.
Également des structures où plusieurs données pas nécessairement du meme type sont regroupés sous une meme variable.

Il existe 6 structures :

- **Date :** Gère le caractère temporel des trajets sous forme de int pour jour/mois/heure
- **Ville :** Regroupe toutes les villes dans un tableau de char* ainsi qu'un int pour le nombre de villes desservies
- **Client :** Contenant l'identifiant (char[]), le mot de passe (char[]), la liste des trajets, le nombre de trajets du client (int)
- **Conducteur:** Contenant l'identifiant (char[]), le mot de passe (char[]), la liste des trajets, le nombre de trajets du conducteur (int)
- **Admin :** Contenant la liste des conducteurs, la liste des clients, le nombre de clients et de conducteurs et un tableau des villes desservies
- **Trajets :** Composé du client, de la date du trajet, du nombre de villes parcourues lors du trajet, et du tableau des villes parcourues

Le squelette du programme se compose de 4 menus :

- **Menu initial :** Permettant de se rendre à n'importe quel autre menu

- **Menu administrateur :** Regroupe les fonctions propres à l'administrateur : Gérer les villes desservies, changer le mot de passe d'un client ou d'un conducteur. Il faut un code d'accès afin de pouvoir accéder à ce dernier

- **Menu conducteur :** Regroupe les fonctions propres au conducteur : S'enregistrer en tant que nouveau conducteur, se connecter à son compte, proposer un trajet, afficher la liste des trajets (passés et à venir) et des clients potentiellement enregistrés sur chaque trajet, modifier son mot de passe

- **Menu client:** Regroupe les fonctions propres au client: s'inscrire, se connecter à son compte, réserver un trajet, annuler un trajet

Fonctions	Rôles
Date constructeurDate (int jour, int mois, int heure)	Prend en entrée des int jour/mois/ heure et permet de renvoyer une date
Villes constructeurVille(){	Construit le tableau des villes desservies
Trajet constructeurTrajet()	Alloue la mémoire pour un trajet
Client constructeurClient()	Alloue la mémoire pour un client
Conducteur constructeurCond()	Alloue la mémoire pour un conducteur
Admin constructeurAdmin()	Alloue la mémoire pour administrer les autres structures
Client setNewClient (char[] prenom, char[] mdp, Admin ad)	Prend en entrée un identifiant, mot de passe et renvoie un nouveau client
Conducteur setNewConducteur (char[] mdp, char[] num_id, Admin ad)	Prend en entrée un identifiant, mot de passe et renvoie un nouveau conducteur
void exporterJsonconducteur (Admin ad)	Permet d'exporter la liste des conducteurs ainsi que leurs trajets au format JSON
int verifId(int id, Admin ad)	Vérifie si le conducteur dont l'identifiant est entré est bien dans la base de données. Renvoie son indice dans la liste des conducteurs, conducteur inconnu sinon
int verifIdCl(char[] idCl, Admin ad)	Vérifie si un client dont l'identifiant est entré est bien dans la base de données. Renvoie son indice dans la liste des clients, client inconnu sinon
void reserver(Trajet trajet, Client c)	Permet à un client de réserver un trajet
Conducteur changeMDP (char[] mdp, Conducteur cond)	Permet au conducteur de changer de mot de passe
void supprimeTrajetPassager (Client client, Trajet trajet)	Permet à un client de supprimer un trajet
int verifMDPcond(char[] mdp, admin)	Vérifie si le mdp du conducteur est correct ou pas lors de la connexion à son compte
Conducteur changeMDP(char[] mdp, Conducteur)	Change le mot de passe du conducteur

Client changeMDPcl(char[] mdp, Client)	Change le mot de passe du client
Void supprimeTrajetPassager(Client, trajet)	Permet à un client de supprimer son trajet
Int danslesborneschiffres(char [])	Vérifie si l'identifiant est bien constitué de chiffres
Char* chiffrement_cesar(char[], int clé)	Permet de chiffrer un mot de passe
void exporterJsonClient(Admin ad)	Permet d'exporter la liste des clients au format JSON
Void importJSON(Admin ad)	Permet d'importer les listes des conducteurs, des clients ainsi que leurs trajets dans un struct admin

B- Mode d'emploi du programme

Dès l'exécution du programme, l'utilisateur est dans le menu initial. Il peut taper :

1 - Pour accéder au menu conducteur

Une fois dans le menu conducteur, l'utilisateur indique par :

1- Si c'est un nouveau conducteur

On lui demande de taper un identifiant (chiffres) et un mot de passe (lettres). Son compte est créé il est automatiquement redirigé vers le menu conducteur.

2- Si c'est un conducteur déjà enregistré

L'utilisateur se connecte en entrant son identifiant et son mot de passe. Une fois validés, il peut modifier des informations concernant son compte en entrant :

1- Pour créer un nouveau trajet

On lui demande combien de villes il souhaite parcourir, de choisir le numéro de la ville parmi la liste des villes pour le départ, les étapes et l'arrivée. Aussi la date et l'horaire du trajet

2- Pour modifier son mot de passe

On lui demande d'entrer un nouveau mot de passe, l'ancien est modifié

3- Pour afficher les trajets passés et à venir

Les dates et horaires de tous les trajets s'affichent

0- Pour quitter le menu conducteur

Il est redirigé vers le menu initial

2- Pour le menu client

Une fois dans le menu client, l'utilisateur indique par :

1- Si c'est un nouveau client

On lui demande de taper un identifiant (lettres) et un mot de passe (lettres) Son compte est créé, il est automatiquement redirigé vers le menu client.

2- Si c'est un client déjà enregistré

L'utilisateur se connecte en entrant son identifiant. Une fois validés, il peut modifier des informations concernant son compte en entrant :

1- Pour réserver un trajet

Il doit choisir le numéro du trajet parmi ceux proposés, la date et l'horaire du trajet

2- Pour afficher ses trajets

Affiche sa liste de trajets

3- Pour supprimer un trajet

Affiche sa liste de trajets numérotés, on lui demande le numéro du trajet à supprimer

3- Pour le menu administrateur

Une fois dans le menu administrateur, un code d'accès (9635) doit etre entré afin d'accéder aux fonctionnalités du menu sauf si on tape 0, on retourne alors au menu initial. Si le code d'accès est bon, l'utilisateur tape :

1- Pour gérer les villes

1- Pour ajouter une ville

Le nom de la ville est tapé, elle est ajoutée à la liste des villes et le nombre de villes augmente d'un.

2- Pour supprimer une ville

L'utilisateur choisit le numéro associé à la ville à supprimer parmi la liste des villes affichées

2- Pour modifier le mot de passe d'un conducteur ou d'un client

1- Pour modifier le mot de passe d'un client

Le nouveau mot de passe est demandé, l'ancien mot de passe est remplacé par le nouveau

2- Pour modifier le mot de passe d'un conducteur

Le nouveau mot de passe est demandé, l'ancien mot de passe est remplacé par le nouveau

C- Difficultés rencontrées

- Les erreurs de segmentation dues à des scanf de char et aux pointeurs étaient certaines des erreurs les plus souvent rencontrées et qui nous ont causé le plus de problèmes dans notre code
- Le format JSON était aussi un problème car nous en avions jamais fait.
- Le chiffrement des mots de passe en utilisant openssl
- L'installation de l'environnement de travail (librairies, ubuntu, visual studio code etc)
- Choisir les variables à utiliser

D- Bilan qualitatif du travail

Au final, nous pensons avoir réussi la plupart des tâches demandées : chiffrer les mots de passe, permettre à de nouveaux conducteurs/clients à s'inscrire, garantir la sécurité lors de l'accès à leurs comptes, vérification des coordonnées (identifiant, mot de passe), changer les mots de passe, réserver, consulter et supprimer des trajets.

Eviter que le programme crash si les données tapées ne sont pas les types attendus, exporter les données sous format JSON. Cependant, nous n'avons pas été en mesure d'importer des données d'un fichier JSON extérieur au programme pour plusieurs causes : problème d'installation de librairie, mais surtout nous pensons que nous avions été un peu trop ambitieux en essayant d'importer des données JSON pour les stocker dans un struct. Ceci dit, nous arrivons assez bien à exporter nos données (client, conducteurs, trajets) en format JSON.

Pour conclure, nous pensons avoir rempli la plupart des tâches demandées.