Ce programme en C simplifie la gestion de répertoire, offrant des fonctionnalités d'ajout, d'affichage, de recherche, et de suppression d'identités.

Rapport de projet C

Programmation en C et C++

Matthieu BRANDAO
Ahmad HOUHOU
Enzo IHADJADENE

PARIS PANTHÉON-ASSAS UNIVERSITÉ

Sommaire

Introduction

Le code fourni implémente un programme de gestion de répertoire qui permet à l'utilisateur d'ajouter, afficher, rechercher et supprimer des enregistrements d'identités. Le programme utilise des structures pour représenter les informations relatives à chaque identité et conserve ces données dans un tableau ainsi qu'un fichier CSV pour l'archivage des données.

Fonctionnement général:

Le programme débute par la définition de deux structures : *Identité* et *Repertoire*. La première représente les informations d'une personne, tandis que la seconde contient un tableau d'identités (*Identités*) et le nombre d'enregistrements dans le répertoire (*enregistrements*). Une instance de la structure *Repertoire* est créée, nommée *Repository*. Ensuite, des fonctions sont définies pour effectuer différentes opérations sur le répertoire, telles que l'ajout, l'affichage, la recherche, et la suppression.

A chaque opération, le fichier CSV est mis à jour directement.

La fonction principale (*main*) utilise une boucle permanente pour afficher un menu à l'utilisateur, lui permettant de choisir parmi différentes fonctionnalités. La boucle continue jusqu'à ce que l'utilisateur choisisse de quitter le programme en saisissant '5'.

Fonctionnalités avec algorithmes explicites pour chaque fonction:

1. **csvToRepository() :**

   - Ouvre le fichier CSV en mode lecture.
   - Parcourt chaque ligne du fichier.
   - Utilise la fonction strtok pour extraire les données séparées par des virgules.
   - Stocke les données dans le tableau du répertoire (Repository).
   - Ferme le fichier après la lecture et met à jour le nombre total d'enregistrements (records) dans le répertoire.

2. **add_to_csv(int i, const char nom[], const char prenom[], const char mail[], const char tel[]) :**

   - Ouvre le fichier CSV en mode ajout.
   - Ajoute une nouvelle ligne au fichier CSV avec les informations fournies (ID, nom, prénom, e-mail, téléphone).
   - Ferme le fichier après l'ajout.

3. **deleteFileContent() :**

   - Ouvre le fichier CSV en mode écriture (ce qui efface le contenu existant).

- Ferme le fichier après l'effacement.

4. **repesitoryToCsv() :**

   - Ouvre le fichier CSV en mode écriture.
   - Parcourt le tableau du répertoire (Repository).
   - Écrit chaque enregistrement dans le fichier CSV.
   - Ajoute une nouvelle ligne après chaque entrée, sauf la dernière.
   - Ferme le fichier après l'écriture.

5. **display_record(int i) :**

   - Affiche les informations relatives à l'identité d'index i dans le répertoire.

6. **create_record() :**

   - Demande à l'utilisateur de saisir les informations pour créer un nouvel enregistrement dans le répertoire.
   - Utilise la fonction add_to_csv pour ajouter le nouvel enregistrement dans le fichier CSV.
   - Incrémente le nombre total d'enregistrements (records) dans le répertoire.

7. **display_directory() :**

   - Affiche l'ensemble du répertoire en utilisant une boucle qui appelle la fonction display_record pour chaque identité dans le répertoire.

8. **search_directory() :**

   - Demande à l'utilisateur de saisir un nom.
   - Recherche et affiche toutes les identités correspondantes dans le répertoire.

9. **delete_record() :**

   - Supprime le contenu du fichier CSV.
   - Demande à l'utilisateur de saisir le nom d'une personne à supprimer.
   - Supprime l'identité correspondante dans le répertoire.
   - Utilise la fonction repesitoryToCsv pour mettre à jour le fichier CSV après la suppression.

Conclusion

Le programme fournit une interface simple pour la gestion d'un répertoire d'identités. Les fonctions permettent à l'utilisateur d'ajouter de nouvelles entrées, d'afficher l'ensemble du répertoire, de rechercher des identités par nom, et de supprimer des entrées. Celui-ci permet aussi l'archivage des données après coupure du programme pour une meilleur expérience utilisateur. Cependant, il convient de noter que le programme pourrait bénéficier d'améliorations, telles que la gestion des erreurs lors de

la saisie des données utilisateur, et une meilleure gestion de la mémoire pour éviter les dépassements de tableau.