Rapport de Mini-Projet C

Programmation C et C++

EFREI L3-APP-LSI2

Matthéo PERELLE,
Aristote ROULLOT

November 16, 2023

Contents

Chapter 1

Présentation du projet

L'objectif de ce projet est de mettre en évidence les connaissances acquises lors du cours de programmation C et C++. Nous avons donc décider de réaliser un répertoire téléphonique dans le quel les données suivantes seront stockées :

- Nom
- Prénom
- Numéro de téléphone
- Adresse mail

Notre programme devra implémenter les fonctionnalités suivantes :

- Ajouter un contact
- Supprimer un contact
- Afficher tous les contacts
- Rechercher un contact
- Quitter le programme

Chapter 2

Implémentation

2.1 Choix de la structure de donnée

Pour réaliser ce projet, nous avons donc dû choisir comment nous allions structurer nos données. Nous avons donc choisi d'utiliser une liste chaînée. Premièrement, cette structure de données nous permet de stocker de façon dynamique sans se soucier du nombre de personnes dans notre répertoire. De plus, cela permet de mettre en avant les connaissance du cours de structure de données que nous avons. Commençons par définir notre structure de données Personne dans laquelle on retrouve le nom, le prénom, le numéro de téléphone et l'adresse mail de la personne. Chaque propriété de la structure est un tableau de caractère de taille 50 et ce même pour le téléphone pour pouvoir stocker les 0 en début de numéro.

```c
/*
 * Structure de donnee Personne
*/
typedef struct Personne {
    char nom[50];
    char prenom[50];
    char numero[50];
    char mail[50];

} Personne;
```

Et voici alors notre structure de base en C de notre liste chaînée:

```c
/*
 * Noeud de la liste chainee
*/
typedef struct RepertoireNode {
    Personne *personne;

    struct RepertoireNode *next;
    struct RepertoireNode *prev;
} RepertoireNode;
```

```c
/*
 * Struture porteuse de la liste chainee
 * permettant de materialiser le debut, et sa taille
*/
struct Repertoire {
    int size;
    RepertoireNode *head;
} typedef Repertoire;
```

Nous avons donc une liste doublement chaînée avec une structure *RepertoireNode* qui contient un pointeur vers une personne et deux pointeurs vers le nœud suivant et précédent. Et une structure *Repertoire* qui contient la taille de la liste et un pointeur vers le premier nœud de la liste.

Chapter 3

Fonctionnalités supplémentaires

Pour ce projet nous avons rajouté quelques fonctionnalités supplémentaires afin de rendre le programme plus agréable à utiliser.

3.1 Validation des entrées

Dans un premier temps, lors de la saisie d'un numéro de téléphone, nous avons mis en place une vérification afin de s'assurer que le numéro entré est bien un numéro de téléphone. Pour ce faire, nous vérifions que le numéro ne contient que des chiffres.

Voici un exemple de la fonction de vérification :

```
Veuillez entrer le numéro de la personne: pasunnumero
numéro invalide, il ne doit contenir que des chiffres et peut commencer par un
↪  +
Veuillez entrer le numéro de la personne: e11993
numero invalide, il ne doit contenir que des chiffres et peut commencer par un
↪  +
Veuillez entrer le numéro de la personne: 07010101011
```

Cette même verification est effectuée pour un email. Nous vérifions que l'adresse mail contient bien un @ et un . afin de s'assurer que l'adresse mail est valide.

```
Veuillez entrer le mail de la personne: pasunemail
mail invalide, il doit contenir au moins 1 point et un @
Veuillez entrer le mail de la personne: pasunemail@
mail invalide, il doit contenir au moins 1 point et un @
Veuillez entrer le mail de la personne: email@email.com
Personne ajoutée avec succès
```

3.2 Gestion d'information identique

Dans l'implémentation de la recherche, l'utilisateur peur entrer un nom, un prénom, un numéro de téléphone ou une adresse mail. Le programme va alors rechercher dans le répertoire

si une personne correspond à la recherche. Si c'est le cas, le programme affiche la personne et si ce n'est pas le cas, le programme affiche un message d'erreur. Il prend aussi en charge avec plusieurs résultat comme par exemple si on recherche "John" et qu'il y a deux personnes qui s'appellent "John", le programme va afficher les deux personnes.

Ce même comportement est utilisé lors de la suppression d'une personne. Si l'utilisateur entre un nom, un prénom, un numéro de téléphone ou une adresse mail et qu'il y a plusieurs personnes qui correspondent à la recherche, le programme va demander à l'utilisateur de choisir la personne qu'il souhaite supprimer.

Notre prend donc bien en charge les informations identiques.

3.3 Modification d'une personne

Nous avons aussi implémenter la possibilité de modifier un contact. Pour se faire l'utilisateur cherche un utilisateur existant par la recherche habituelle, puis peut modifier ou non chaque champs. Voici un exemple.

```
Veuillez entrer une information quelconque de la personne pour la rechercher
 ↪  (vide pour annuler): perelle
Vous allez modifier la personne suivante :
========================================
Nom: perelle
Prenom: mattheo
Numero: 8551
Mail: @.
========================================
Veuillez entrer le nouveau nom de la personne (vide pour ne pas modifier):
 ↪  PERELLE
Veuillez entrer le nouveau prenom de la personne (vide pour ne pas modifier):
Veuillez entrer le nouveau numero de la personne (vide pour ne pas modifier):
 ↪  9874
Veuillez entrer le nouveau mail de la personne (vide pour ne pas modifier):
 ↪  mattheo.perelle@efrei.net
Voici la personne modifiée :
========================================
Nom: PERELLE
Prenom: mattheo
Numero: 9874
Mail: mattheo.perelle@efrei.net
========================================

Entrer (y) pour confirmer la modification : y
Personne modifiée avec succès
```

3.4 Sauvegarde et chargement

Nous avons aussi ajouté la possibilité de sauvegarder et charger le répertoire dans un fichier. Pour ce faire, nous avons utilisé la fonction *fopen* de la librairie *stdio.h* afin d'ouvrir un fichier en mode écriture ou lecture. Nous avons ensuite utilisé la fonction *fprintf* afin d'écrire dans le fichier et *fscanf* afin de lire dans le fichier.

Le chargement du fichier se fait au lancement du programme, et la sauvegarde se fait lors de la fermeture du programme.

Les données sont stockées dans le fichier de la façon suivante en suivant les normes CSV (semicolon-separated) où chaque ligne correspond à une personne et chaque propriété est séparée par un point-virgule :

```
John;Doe;john.doe@email.com;139123809
Piere;mary;pierre.mary.g@efrei.net;078614514
```

Chapter 4

Lancement du projet

Pour lancer le projet sur mac ou linux, il suffit d'utiliser le Makefile. Celui-ci permet de compiler le projet et de lancer le programme. Pour ce faire, il suffit d'utiliser la commande suivante :

```
make run
```

Sur windows il suffit de compiler le projet avec la commande suivante :

```
gcc .\src\core\personne.c .\src\core\personne.h .\src\core\repertoire.c
↪  .\src\core\repertoire.h .\src\storage\storage_csv.c
↪  .\src\storage\storage_csv.h .\src\utils\utils.c .\src\utils\utils.h
↪  .\src\validators\validators.c .\src\validators\validators.h
↪  .\src\commands\commands.c .\src\commands\commands.h .\src\main.c -o main
```

puis de lancer le programme avec la commande suivante :

```
.\main
```