

Rapport de Mini-Projet C
Programmation C et C++
EFREI L3-APP-LSI2

Matthéo PERELLE,
Aristote ROULOU

November 16, 2023

Contents

1	Présentation du projet	2
2	Implémentation	3
2.1	Choix de la structure de donnée	3
3	fonctionnalités supplémentaires	5
3.1	Validation des entrées	5
3.2	Sauvegarde et chargement	6
4	Lancement du projet	7

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 décidé de réaliser un répertoire téléphonique dans le quel les données suivantes seront stockées :

- Nom
- Prénom
- Numéro de téléphone
- Adresse mail

Notre programme devra implementer 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 du choisir comment nous allons structurer nos données. Nous avons donc choisi d'utiliser une liste chaînée. Premièrement, cette structure de donnée 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 connaissances du cours de structure de donnée que nous avons.

Commençons par définir notre structure de donnée *Personne* dans laquelle on retrouvera le nom, le prénom, le numéro de téléphone et l'adresse mail de la personne. Chaque propriété de la structure est un tableau de caractères de taille 50 et ceux même pour le téléphone pour pouvoir stocker les 0 en début de numéro.

```
/*  
 * Structure de donnée 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:

```
/*  
 * Noeud de la liste chaînée  
 */  
typedef struct RepertoireNode {  
    Personne *personne;  
  
    struct RepertoireNode *next;  
    struct RepertoireNode *prev;
```

```

} RepertoireNode;

/*
 * Structure 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 noeud suivant et précédent. Et une structure *Repertoire* qui contient la taille de la liste et un pointeur vers le premier noeud de la liste.

Chapter 3

fonctionnalités supplémentaires

Pour ce projet nous avons rajoutés 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 :

```
Veillez entrer le numero de la personne (pas d'indicatif, seulement des
↪ chiffres): pasunnumero
numero invalide, il ne doit contenir que des chiffres et peut commencer par un
↪ +
Veillez entrer le numero de la personne (pas d'indicatif, seulement des
↪ chiffres): e11993
numero invalide, il ne doit contenir que des chiffres et peut commencer par un
↪ +
Veillez entrer le numero de la personne (pas d'indicatif, seulement des
↪ chiffres): 07010101011
```

Cette même vérification 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.

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

3.2 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 ce fait au lancement du programme, et la sauvegarde ce 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\main.c .\src\commands\commands.c .\src\commands\commands.h  
↪ .\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 -o main  
./main
```