

Programmation séquentielle

Série 7 - Base d'un annuaire

01.11.2022

Buts

- Utilisation de structs.
- Implémentation d'une structure de données.
- Implémentation d'une API.
- Modularisation.
- Gestion des erreurs.

Introduction

Le but de cet exercice est de développer des fonctions permettant de manipuler un annuaire. La définition des types nécessaires vous est donné et vous devez développer les fonctions liées à ces types. Ce travail servira de base pour le prochain qui sera un TP noté. Veuillez donc à le commencer suffisamment tôt et à avoir un code original (créé par vous).

Énoncé

Types utilisés

L'annuaire stocke pour chaque personne :

- Le prénom sous forme de chaîne de caractères;
- Le nom sous forme de chaîne de caractères;
- La date de naissance sous forme d'un type structuré;
- Le numéro de téléphone sous forme d'un nombre entier ¹.

¹Ce n'est évidemment pas la représentation idéale pour ce type d'information puisqu'il n'est pas possible de représenter les 0 au début du numéro. Mais cela conviendra pour l'exercice.

Ainsi, une date est représentée par le type

```
typedef struct {
    int day, month, year;
} date_t;
```

et une personne par le type ²

```
typedef struct {
    char name[40];
    char surname[40];
    date_t birthday;
    int phone_number;
} person_t;
```

L'annuaire est représenté sous forme d'un tableau statique de personnes. Comme on travaille avec un tableau statique, sa capacité est fixe. La capacité de l'annuaire est définie à l'aide d'une macro DIR_CAPACITY (par exemple #define DIR_CAPACITY 1000). Le nombre d'éléments stockés dans l'annuaire peut cependant évoluer au fil de l'exécution du programme, ainsi, la taille réelle est stockée dans une variable liée au tableau. Un annuaire est représenté par le type :

```
typedef struct {
    int size;
    person_t content[DIR_CAPACITY];
} directory_t;
```

Fonctions à implémenter

Le but est d'implémenter les fonctions de traitement d'annuaire suivantes :

```
// Prints a person record with the format "Name Surname, day-month-year, phone number"
// Parameter : p - the person
void person_print(person_t p);

// Prints the content of a directory each line with the format "[n] person"
// with n being the position in the directory
// Parameters : dir - the directory
void dir_print(directory_t *dir);

// Initialize a directory by setting its size to 0 and thus actually erasing its content
// Parameter : dir - the directory
void dir_init(directory_t *dir);

// Add a person to the directory if its capacity is not reached and increase its size
// Parameters : dir - the directory, p - the person
// Returns : 0 if success, 1 otherwise
int dir_add_entry(directory_t *dir, person_t p);

// Fill a directory with content, erasing any previous content
// Parameter : dir - the directory
// Returns : 0 if success, 1 otherwise
int dir_fill(directory_t *dir);

// Delete the entry at position n of the directory if n is smaller than
// the size of the directory, does nothing otherwise
// Parameters : dir - the directory, n - the position
void dir_delete_entry(directory_t *dir, int n);
```

²La limite de cette représentation est que chaque personne peut avoir un nom et un prénom d'au plus 40 caractères.

Vous devez respecter la signature des fonctions. Chaque fonction possède un commentaire de documentation définissant ce que doit faire la fonction. Le nombre entier retourné par certaines fonctions est un code d'erreur qui est 0 en cas de succès et 1 en cas d'échec (par exemple si la capacité de l'annuaire est atteinte).

Vous remarquez que l'annuaire est systématiquement passé par référence afin de permettre sa modification et d'éviter de copier l'intégralité de son contenu dans les fonctions d'affichage.

On présente ici une proposition d'implémentation pour la fonction `dir_fill`

```
// Fill a directory with content, erasing any previous content
// Parameter : dir - the direcorey
// Returns : 0 if success, 1 otherwise
int dir_fill(directory_t *dir){
    dir_init(dir);
    if(dir_add_entry(dir, (person_t){ "Jean", "Paul", (date_t){18, 12, 1967}, 12346890})) return 1;
    if(dir_add_entry(dir, (person_t){ "Arthur", "Rimbaud", (date_t){1, 1, 2002}, 73645})) return 1;
    if(dir_add_entry(dir, (person_t){ "Paul", "Albuquerque", (date_t){23, 6, 1902}, 28736474})) return 1;
    if(dir_add_entry(dir, (person_t){ "Christophe", "Charpilloz", (date_t){11, 4, 2008}, 76253})) return 1;
    if(dir_add_entry(dir, (person_t){ "Emily", "Dickinson", (date_t){19, 8, 1950}, 636363})) return 1;
    if(dir_add_entry(dir, (person_t){ "Ada", "Lovelace", (date_t){12, 12, 2022}, 12345})) return 1;
    if(dir_add_entry(dir, (person_t){ "Qwertz", "Uiop", (date_t){1, 1, 562}, 67879})) return 1;
    return 0;
}
```

Après l'appel à cette fonction, l'appel à `dir_print` devra produire :

```
[0] Jean Paul, 18-12-1967, 12346890
[1] Arthur Rimbaud, 1-1-2002, 73645
[2] Paul Albuquerque, 23-6-1902, 28736474
[3] Christophe Charpilloz, 11-4-2008, 76253
[4] Emily Dickinson, 19-8-1950, 636363
[5] Ada Lovelace, 12-12-2022, 12345
[6] Qwertz Uiop, 1-1-562, 67879
```

Programme principal

Ecrivez un programme qui utilise vos fonctions. Votre programme devra au minimum :

- Initialiser un annuaire;
- Peupler l'annuaire de données, et afficher un message d'erreur et terminer si la capacité est atteinte³;
- Afficher l'annuaire;
- Supprimer un élément de l'annuaire (qui ne soit pas le dernier);
- Afficher à nouveau l'annuaire après la suppression.

³Essayez de définir `DIR_CAPACITY` à une valeur petite, par exemple 5, pour vérifier que le comportement est le bon lorsque la taille maximale est atteinte.