UNIVERSITÉ DE PAU ET DES PAYS DE L'ADOUR | COLLÈGE STEE SCIENCES ET TECHNOLOGIES POUR L'ÉNERGIE ET L'ENVIRONNEMENT

# Techniques de programmation - TP nº 2 - Types complexes et fonctions du langage C

Samson Pierre <samson.pierre@univ-pau.fr>

03/10/2021

Ce TP a pour objectif de vous familiariser avec les types complexes et fonctions du langage C. Pour chaque exercice, sauf indication contraire, aucune fonction n'est autorisée et les mêmes options de compilation que celles du cours doivent être utilisées.

## Exercice nº 1

Créez le programme `library.c` qui contient une structure `library_s` qui décrit une bibliothèque. Une bibliothèque est caractérisée par son nom `name` (un tableau de caractères de taille 100), ses jours d'ouverture `days` (un tableau d'énumérations `day_e` de taille 7), son nombre de jours d'ouverture `ndays` (un entier qui est le nombre d'éléments du tableau `days`), ses livres `books` (un tableau de structures `book_s` de taille 100) et son nombre de livres `nbooks` (un entier qui est le nombre d'éléments du tableau `books`). Ajoutez donc ces cinq membres à la structure `library_s`. Un jour est soit lundi `monday`, soit mardi `tuesday`, soit mercredi `wednesday`, soit jeudi `thursday`, soit vendredi `friday`, soit samedi `saturday`, soit dimanche `sunday`. Ajoutez donc ces sept valeurs à l'énumération `day_e`. Un livre est caractérisé par son nom `name` (un tableau de caractères de taille 100), ses auteurs `authors` (un tableau à deux dimensions de tailles 10 et 100), son nombre d'auteurs `nauthors` (un entier qui est le nombre d'éléments dans le tableau `authors`), son éditeur `publisher` (un tableau de caractères de taille 100), son année `year` (un entier) et son ISBN `isbn` (un entier). Ajoutez donc ces six membres à la structure `book_s`.

Créez les bibliothèques « Sciences Library » et « Novel Library ». La bibliothèque « Sciences Library » est ouverte du lundi au jeudi. Elle contient les livres « The C Programming Language », écrit par Brian W. Kernighan et Dennis M. Ritchie, édité par Prentice Hall en 1988 avec l'ISBN 9780131103627 et « C: The Complete Reference », écrit par Herbert Schildt, édité par McGraw-Hill Education en 2000 avec l'ISBN 9780072121247. Ajoutez donc ces deux livres à cette bibliothèque. La bibliothèque « Novel Library » est ouverte du mardi au vendredi. Elle contient les livres « Harry Potter and the Philosopher's Stone », écrit par J. K. Rowling, édité par Bloomsbury en 1997 avec l'ISBN 9780747532699 et « Harry Potter and the Chamber of Secrets », écrit par J. K. Rowling, édité par Bloomsbury en 1998 avec l'ISBN 9780747538493. Ajoutez donc ces deux livres à cette bibliothèque.

Créez une fonction `book_print` qui prend en paramètre un livre `book` et qui ne retourne rien. Cette fonction affiche les informations sur `book`. Créez une fonction `library_print` qui prend en paramètre une bibliothèque `library` et qui ne retourne rien. Cette fonction affiche les informations sur `library` et appelle `book_print` sur tous les livres de `library`. Appelez la fonction `library_print` sur vos deux bibliothèques.

Voici le résultat attendu :

```
$ ./library.out
("Sciences Library", ("monday", "tuesday", "wednesday", "thursday"), (("The C Programming
Language", ("Brian W. Kernighan", "Dennis M. Ritchie"), "Prentice Hall", 1998, 9780131103627), ("C:
The Complete Reference", ("Herbert Schildt"), "McGraw-Hill Education", 2000, 9780072121247)))
("Novel Library", ("tuesday", "wednesday", "thursday", "friday"), (("Harry Potter and the
Philosopher's Stone", ("J. K. Rowling"), "Bloomsbury", 1997, 9780747532699), ("Harry Potter and the
Chamber of Secrets", ("J. K. Rowling"), "Bloomsbury", 1998, 9780747538493)))
$
```

Voici les fonctions autorisées pour cet exercice :

```
int printf(const char *format, ...);
```

## Exercice nº 2

Reprenez le code de l'exercice précédent. Définissez le type `library_t` à partir du type existant `struct library_s`, le type `book_t` à partir du type existant `struct book_s` et le type `day_t` à partir du type existant `enum day_e`. Utilisez le type `library_t` à la place de `struct library_s`, le type `book_t` à la place de `struct book_s` et le type `day_t` à la place du type `enum day_e` afin d'alléger votre code.

Voici les fonctions autorisées pour cet exercice :

```
int printf(const char *format, ...);
```

## Exercice n° 3

Reprenez le code de l'exercice précédent. Au lieu de définir le type `library_t` à partir du type existant `struct library_s`, définissez-le à partir d'une structure anonyme décrivant une bibliothèque. Au lieu de définir le type `book_t` à partir du type existant `struct book_s`, définissez-le à partir d'une structure anonyme décrivant un livre. Au lieu de définir le type `day_t` à partir du type existant `enum day_e`, définissez-le à partir d'une énumération anonyme décrivant un jour. Dorénavant, les types `struct library_s`, `struct book_s` et `enum day_e` n'existent plus dans votre programme.

Voici les fonctions autorisées pour cet exercice :

```
int printf(const char *format, ...);
```

## Exercice n° 4

Reprenez le code de l'exercice précédent. Remplacez les membres de type tableau dans la structure décrivant une bibliothèque par des pointeurs. Remplacez les membres de type tableau dans la structure décrivant un livre par des pointeurs. Ainsi, il n'y a plus de limite concernant la taille de ces membres (si ce n'est la mémoire maximale de la machine).

Créez une fonction `book_create` sans paramètre qui retourne un pointeur livre. Cette fonction effectue une allocation dynamique de mémoire pour un livre et initialise ses membres `name` à `NULL`, `authors` à `NULL`, `nauthors` à `0`, `publisher` à `NULL`, `year` à `0` et `isbn` à `0`. Si c'est un succès, cette fonction retourne le pointeur livre. Si c'est un échec, cette fonction retourne `NULL`. Créez une fonction `library_create` sans paramètre qui retourne un pointeur bibliothèque. Cette fonction effectue une allocation dynamique de mémoire pour une bibliothèque et initialise ses membres `name` à `NULL`, `days` à `NULL`, `ndays` à `0`, `books` à `NULL` et `nbooks` à `0`. Si c'est un succès, cette fonction retourne le pointeur bibliothèque. Si c'est un échec, cette fonction retourne `NULL`.

Créez une fonction `book_free` qui prend en paramètre un pointeur livre `book` et qui ne retourne rien. Cette fonction libère la mémoire allouée dynamiquement pour `book`. Créez une fonction `library_free` qui prend en paramètre un pointeur bibliothèque `library` et qui ne retourne rien. Cette fonction libère la mémoire allouée dynamiquement pour `library`.

Créez une fonction `book_add_author` qui prend en paramètres un pointeur livre `book`, une chaîne de caractères `author` et qui retourne un entier. Cette fonction ajoute `author` à `book` en réallouant dynamiquement la mémoire. Si c'est un succès, cette fonction retourne 0. Si c'est un échec, cette fonction retourne -1. Créez une fonction `library_add_day` qui prend en paramètres un pointeur bibliothèque `library`, un jour `day` et qui retourne un entier. Cette fonction ajoute `day` à `library` en réallouant dynamiquement la mémoire. Si c'est un succès, cette fonction retourne 0. Si c'est un échec, cette fonction retourne -1. Créez une fonction `library_add_book` qui prend en paramètres un pointeur bibliothèque `library`, un livre `book` et qui retourne un entier. Cette fonction ajoute `book` à `library` en réallouant dynamiquement la mémoire. Si c'est un succès, cette fonction retourne 0. Si c'est un échec, cette fonction retourne -1.

Lancez votre programme avec la commande `valgrind` afin de vérifier si la mémoire allouée dynamiquement a correctement été libérée.

Voici les fonctions autorisées pour cet exercice :

```
void free(void *ptr);
void *malloc(size_t size);
int printf(const char *format, ...);
void *realloc(void *ptr, size_t size);
```

## Exercice n° 5

Créez un programme `mirror.c` qui lit un paramètre passé en ligne de commande. Ce paramètre est la chaîne de caractères pour laquelle il faut créer le miroir. Affichez une erreur dans le flux d'erreur standard et retournez le code d'erreur 1 si le nombre de paramètres passés en ligne de commande est différent de un ou si le paramètre passé en ligne de commande a une longueur supérieure à 10.

Créez une fonction `mirror` qui prend en paramètres une chaîne de caractères constante `str`, une chaîne de caractères `res` et ne retourne rien. Cette fonction crée le miroir de `str` et stocke le résultat dans `res`. Passez à `str` le paramètre passé en ligne de commande. Passez à `res` un tableau de caractères de taille 11. Affichez le résultat dans le flux de sortie standard.

Voici le résultat attendu :

```
$ ./mirror.out
invalid number of arguments
$ echo $?
1
```

```
$ ./mirror.out hello bonjour
invalid number of arguments
$ echo $?
1
$ ./mirror.out programming
the string length is greater than 10
$ echo $?
1
$ ./mirror.out hello
"olleh"
$ echo $?
0
$
```

Voici les fonctions autorisées pour cet exercice :

```
int fprintf(FILE *stream, const char *format, ...);
int printf(const char *format, ...);
size_t strlen(const char *s);
```

## Astuces

Pour l'exercice n° 5, vous devez utiliser des paramètres passés en ligne de commande. Le passage de paramètres en ligne de commande sera présenté dans le CM n° 3. Pour récupérer les paramètres passés en ligne de commande, vous devez utiliser la signature à deux paramètres de la fonction `main` :

```
int main(int argc, char *argv[]);
```

Le paramètre `argc` contient le nombre de paramètres passés en ligne de commande. Le paramètre `argv` contient les paramètres passés en ligne de commande.