Programmation C
TP 5

Des tableaux d'entiers positifs de taille variable et un tri fusion.

Encore des tableaux... Mais ici, comme pour les chaînes de caractères, nous allons utiliser des tableaux dont on peut retrouver la taille. La bibliothèque `string.h` contient une fonction `strlen` qui retourne la longueur d'une chaîne de caractères en C grâce au caractère de fin de chaîne : `\0`. On ne peut pas connaître la taille des tableaux en C mais en considérant des tableaux d'entiers positifs et en utilisant le chiffre -1 comme "chiffre de fin de tableaux", on va pouvoir produire un comportement analogue.

Exercice 1 Premier pas avec ces tableaux

Télécharger depuis la plate-forme d'enseignement en ligne une ébauche `main.c` pour ce TP. Cette ébauche contient deux fonctions : une pour fabriquer un nouveau tableau en mémoire et une autre pour libérer la mémoire utilisée par un tableau précédemment alloué. Une compréhension fine de ces deux fonctions n'est pas nécessaire, il est important de comprendre ce qu'elles font par contre.

- Implanter une fonction `int array_size(int* array)` qui retourne la longueur d'un tableau d'entier.
- Implanter une fonction `void print_array(int* array)` qui affiche le contenu d'un tableau sur la sortie standard.
- Implanter une fonction `int are_arrays_equal(int* first, int* second)` qui retourne 1 si les deux tableaux ont le même contenu et 0 sinon.
- Implanter une fonction `int* copy_array(int* array)` qui retourne une nouvelle copie identique du tableau donné en argument. Attention, la copie doit utiliser une nouvelle place mémoire, il ne s'agit pas ici de retourner exactement l'argument de la fonction.

Exercice 2 Fabrication de tableaux

Plusieurs méthodes peuvent permettre d'obtenir de nouveaux tableaux. Parmi ces dernières, demander à l'utilisateur des nombres ou bien utiliser la génération aléatoire de l'ordinateur.

- Implanter une fonction `int* fill_array(void)` qui demande à l'utilisateur d'entrer en console une longueur puis des entiers positifs. Retourner ensuite un tableau fraîchement alloué contenant les entiers donnés par l'utilisateur.
- Implanter une fonction `int* random_array(int size, int max_entry)` qui retourne un nouveau tableau contenant `size` entiers positifs compris entre `0` et `max_entry`.
- Implanter une fonction `int* concat_array(int* first, int* second)` qui prend en argument deux tableaux et retourne un nouveau tableau contenant les entrées concaténées des deux tableaux.

Exercice 3 Un tri fusion verbeux

On essaie maintenant d'implanter un tri fusion en utilisant les fonctions précédentes. On voudrait obtenir possiblement un tri qui affiche tout ce qu'il fait car globalement, le tri fusion reste compliqué dans son traçage d'exécution.

- Implanter une fonction `int* merge_sorted_arrays(int* first, int* second)` qui retourne un tableau trié dont les entrées sont exactement celles des deux tableaux en argument. Ces deux tableaux en arguments sont supposés déjà triés.
- Implanter une fonction `void split_arrays(int* array, int** first, int** second)` qui prend en argument un tableau et deux pointeurs vers des tableaux. Cette fonction doit fabriquer deux tableaux (qui seront placés dans les pointeurs) contenant chacun la moitié des éléments du tableau `array` en argument. Si `array` contient un nombre impair de nombre, un des deux tableaux fabriqués aura un élément de plus. Une solution raisonnable consiste à placer les premiers éléments dans `(*first)` et le reste dans `(*second)`.
- Implanter une fonction `int* merge_sort(int* array)` qui produit un nouveau tableau trié contenant les mêmes éléments que le tableau `array` en argument. Cette fonction devra utiliser l'algorithme du tri fusion (voir Wikipedia par exemple). Cette fonction sera donc récursive. N'hésitez pas à faire des affichages à chaque division et à chaque fusion pour obtenir un traçage de votre algorithme. Il suit un exemple de traçage.

Ce TP est à rendre sur la plate forme en ligne Moodle ou e-learning.

```
Now a random array of length 20 whose entries are lower than 100 :
19 - 44 - 95 - 57 - 28 - 13 - 94 - 44 - 61 - 87 - 24 - 0 - 2 - 16 - 5 - 9 - 32 - 3 - 45 - 9
Using a merge sort, we get :
Split array in two part :
19 - 44 - 95 - 57 - 28 - 13 - 94 - 44 - 61 - 87 - 24 - 0 - 2 - 16 - 5 - 9 - 32 - 3 - 45 - 9
19 - 44 - 95 - 57 - 28 - 13 - 94 - 44 - 61 - 87
24 - 0 - 2 - 16 - 5 - 9 - 32 - 3 - 45 - 9
Split array in two part :
19 - 44 - 95 - 57 - 28 - 13 - 94 - 44 - 61 - 87
19 - 44 - 95 - 57 - 28
13 - 94 - 44 - 61 - 87
Split array in two part :
19 - 44 - 95 - 57 - 28
19 - 44
95 - 57 - 28
Split array in two part :
95 - 57 - 28
95
57 - 28
Merge the two following ones :
95
28 - 57
28 - 57 - 95
Merge the two following ones :
19 - 44
28 - 57 - 95
19 - 28 - 44 - 57 - 95
Split array in two part :
13 - 94 - 44 - 61 - 87
13 - 94
```

```
44 - 61 - 87
Split array in two part :
44 - 61 - 87
44
61 - 87
Merge the two following ones :
44
61 - 87
44 - 61 - 87
Merge the two following ones :
13 - 94
44 - 61 - 87
13 - 44 - 61 - 87 - 94
Merge the two following ones :
19 - 28 - 44 - 57 - 95
13 - 44 - 61 - 87 - 94
13 - 19 - 28 - 44 - 44 - 57 - 61 - 87 - 94 - 95
Split array in two part :
24 - 0 - 2 - 16 - 5 - 9 - 32 - 3 - 45 - 9
24 - 0 - 2 - 16 - 5
9 - 32 - 3 - 45 - 9
Split array in two part :
24 - 0 - 2 - 16 - 5
24 - 0
2 - 16 - 5
Split array in two part :
2 - 16 - 5
2
16 - 5
Merge the two following ones :
2
5 - 16
2 - 5 - 16
Merge the two following ones :
0 - 24
2 - 5 - 16
0 - 2 - 5 - 16 - 24
Split array in two part :
9 - 32 - 3 - 45 - 9
9 - 32
3 - 45 - 9
Split array in two part :
3 - 45 - 9
3
45 - 9
Merge the two following ones :
3
9 - 45
3 - 9 - 45
Merge the two following ones :
9 - 32
3 - 9 - 45
3 - 9 - 9 - 32 - 45
Merge the two following ones :
0 - 2 - 5 - 16 - 24
3 - 9 - 9 - 32 - 45
0 - 2 - 3 - 5 - 9 - 9 - 16 - 24 - 32 - 45
Merge the two following ones :
13 - 19 - 28 - 44 - 44 - 57 - 61 - 87 - 94 - 95
0 - 2 - 3 - 5 - 9 - 9 - 16 - 24 - 32 - 45
0 - 2 - 3 - 5 - 9 - 9 - 13 - 16 - 19 - 24 - 28 - 32 - 44 - 44 - 45 - 57 - 61 - 87 - 94 - 95
```

```
Here is the sorted array :
0 - 2 - 3 - 5 - 9 - 9 - 13 - 16 - 19 - 24 - 28 - 32 - 44 - 44 - 45 - 57 - 61 - 87 - 94 - 95
```