Langace C - Travaux Dirigés n^o 5

— Licence Informatique —

En C, un tableau est simplement un pointeur vers une zone mémoire. Il n'est pas possible de retrouver la taille d'un tableau à partir de ce pointeur. De plus, lors d'un accès à un élément, aucune vérification n'est faite sur les bornes. Le but de cette séance est de définir une structure permettant de manipuler des tableaux qui "connaissent" leur taille. Pour simplifier, on traitera uniquement des tableaux d'entiers.

On définit une structure `array` contenant une taille et un pointeur vers une zone mémoire de la façon suivante.

```
typedef struct {
  int* tab;
  int size;
} Array;
```

▶ Exercice 1 (Allocation et libération)

1. Écrire une fonction `arrayAlloc` qui alloue une structure `Array`. Cette fonction prendra en paramètre la taille du tableau à allouer. Cette fonction devra pouvoir être utilisée de la façon suivante :

   ```
   Array a = arrayAlloc(10);
   ```

2. Écrire une fonction `arrayFree` prenant en paramètre l'adresse d'une structure `Array` et qui libère l'espace mémoire occupé par le tableau de cette structure.

▶ Exercice 2 (Accès aux éléments)

Écrire des fonctions `get` et `set` permettant de récupérer ou de modifier la valeur d'un des éléments du tableau. Ces fonctions prendront en paramètre le tableau et l'indice de l'élément. Elles vérifieront que l'indice est correct. La fonction `set` prendra en outre la nouvelle valeur à assigner à l'élément. Le tableau peut-il être passé directement ou par l'intermédiaire d'un pointeur ? Quels sont les avantages et les inconvénients des deux méthodes ?

▶ Exercice 3 (Accès aux éléments (suite))

1. Regrouper les deux fonctions `get` et `set` de l'exercice précédent en une seule fonction `arrayAccess` qui retournera un pointeur sur l'élément du tableau. Ceci permettra donc de l'utiliser des deux manières suivantes

   ```
   *arrayAccess(a, i) = 5;              // Modification de la valeur
   printf("%d\n", *arrayAccess(a, i)); // Accès à la valeur
   ```

2. Reécrire cette fonction en utilisant des références (sur la structure `Array` en argument et sur la valeur de retour) plutôt que des pointeurs. Quel est l'intérêt de cette nouvelle écriture ?

▶ Exercice 4 (Concaténation de tableaux)

Écrire une fonction `arrayCat` qui concatène deux tableaux. La fonction prendra en argument les adresses de deux structures `Array` et renverra la structure. Peut-on écrire cette fonction en utilisant un simple passage par variables ? Quels sont les avantages et les inconvénients des deux méthodes ?

▶ **Exercice 5 (Échange de tableaux)**
Écrire une fonction `arraySwap` qui échange les contenus de deux tableaux. Cette fonction prendra en argument les adresses de deux structures `Array`. Réécrire cette fonction en utilisant des références ? Comment s'utilise-t-elle ? Peut-on écrire cette fonction en utilisant un simple passage par variables ? Quels sont les avantages et les inconvénients des différentes méthodes ?