Chapitre 9

Tableaux et fonctions

9.1 Tableaux comme paramètres de fonctions

On peut passer un tableau en paramètre d'une fonction. Pour spécifier dans l'en-tête d'une fonction qu'un paramètre est de type tableau, on ajoute après le nom du paramètre les symboles `[]`. Ainsi, si l'on souhaite définir comme paramètre un tableau d'entiers appelé `tab`, le paramètre qui apparait dans la définition de la fonction sera `int tab[]`.

Attention : Dans l'en-tête, il ne faut absolument rien mettre entre les crochets. Et lors d'un appel de la fonction, la valeur du paramètre de type tableau sera le nom de la variable (de type tableau) sans les crochets !

Ainsi, les deux définitions suivantes ont exactement la même signification :

```
double f(double tab[1000]) { ... }
double f(double tab[]) { ... }
```

Si l'on a besoin de connaître le nombre d'éléments du tableau (pour le parcourir par exemple), il faut alors passer un deuxième paramètre de type `int` à cette fonction. À l'appel, sa valeur sera le nombre d'éléments stockés dans le tableau passé en paramètre.

```
double f(double tab[], int nbElements)
{
  int i=0;
  while(i<nbElements)
  {
   ...
   i++;
  }
}
```

On appellera $f(t, 5)$ si l'on sait que t est un tableau à `5` éléments. Le corps de la fonction `f` pourra ainsi utiliser la valeur de `m` pour accéder aux cases `t[i]` pour $0 \leq i \leq nbElements - 1$ du tableau `tab`.

Par exemple la fonction `afficheTableau` définie ci-dessous reçoit en paramètre l'adresse du tableau et son nombre d'éléments.

```
void afficheTableau(int tab[], int nbElements)
{
  int i = 0;
  while (i<nbElements)
  {
    affichage(tab[i], " ");
    i++;
```

```
8    }
9     affichage("\n");
10 }
11
12 int main()
13 {
14   int tabint1[] = {3, 7, 2, 6, 1, 45, 23} ;
15   int tabint2[] = {23, 5, 40, 8, -2, 39, -31,25, 62, 12 };
16   int longtab1 = 7 ; // longueur de tabint1
17   int longtab2 = 10 ; // longueur de tabint2
18   afficheTableau(tabint1, longtab1) ;
19   afficheTableau(tabint2, longtab2) ;
20   return 0;
21 }
```

Ainsi, l'exécution du code affichera à l'écran :

```
3  7  2  6  1  45  23
23  5  40  8  -2  39  -31  25  62  12
```

▶ Sur ce thème : **Exercices 1 et 2 à 4 du TD 9**

9.2 Portée des modifications

Lorsque l'on passe un tableau en paramètre d'une fonction, les modifications des éléments du tableau effectuées par les instructions de la fonction sont répercutées dans le reste du programme.

En effet *le tableau est un argument qui est toujours passé par valeur.* Or, la valeur d'un tableau est l'adresse du bloc de mémoire alloué ; c'est donc cette adresse qui sert à initialiser le paramètre correspondant, lors de l'invocation de la fonction. Comme seule l'adresse du tableau, et non son contenu, est copiée lors de l'invocation de la fonction, celle-ci opère sur le tableau lui-même, et non sur une copie.

Avec ce passage par adresse "implicite", on peut modifier la valeur d'un élément du tableau dans la fonction. Par exemple, on échange la valeur de deux éléments d'un tableau.

À retenir : Du point de vue pratique, on peut modifier les éléments d'un tableau passé en paramètre.

```
1  void incremente_tableau(int tab[], int nbElements)
2  {
3         int i = 0;
4         while (i< nbElements)
5         {
6          adrtab[i]++;
7          i=i+1;
8         }
9  }
10
11 int main()
12 {
13   int tabint1[] = {3, 7, 2, 6, 1, 45, 23} ;
14   int longtab1 = 7 ; // longueur de tabint1
15   incremente_tableau(tabint1, longtab1);
16   afficheTableau(tabint1, longtab1) ;
17   return(0);
18 }
```

Ainsi, l'exécution du code affichera à l'écran :

```
4  8  3  7  2  46 24
```

▶ Sur ce thème : **Exercice 5 du TD 9**

TD9 : Tableaux et fonctions

✓ Exercice 1 : Fonction et passage de tableau*

Définir une fonction `somme` prenant en paramètre un tableau d'entiers et retournant la somme des nombres de ce tableau. Dans le programme principal, affecter à la variable `tab` le tableau `5,6,8,14,17`. Afficher ensuite la somme des valeurs des éléments du tableau à l'aide de la fonction `somme`.

® Exercice 2 : Zéros d'un tableau

Écrire une fonction qui retourne le nombre de zéros contenus dans un tableau d'entiers.

✓ Exercice 3 : Égalité de tableaux *

Bien souvent, on veut comparer deux tableaux sur la base de leur contenu. Deux tableaux `t1` et `t2` seront considérés comme égaux si et seulement si ils ont la même longueur et les éléments de même indice sont égaux, c'est à dire que `t1[i] == t2[i]` pour tout indice i.

Écrire une fonction appelée `estEgal` qui réalise et retourne ce test d'égalité pour des tableaux de type `int`.

® Exercice 4 : Éléments relativement triés

Soit un tableau d'entier. Écrire une fonction qui renvoie le nombre des éléments qui sont supérieurs ou égaux à leur successeur dans le tableau.

✓ Exercice 5 : Saisie et traitement d'un tableau de notes**

Le but de cet exercice est d'écrire un programme qui permet à un utilisateur de saisir les notes d'un étudiant et de les stocker dans un tableau. La saisie s'arrête lorsque l'utilisateur a saisi un nombre strictement inférieur à 0 ou strictement supérieur à 20. Le programme affiche ensuite le minimum, le maximum et la moyenne des notes.

Question 5.1 : Écrire une fonction `afficheTableau` permettant d'afficher un tableau de nombre réels et le programme principal permettant de la tester.

Question 5.2 : Définir une fonction `moyenneTab` prenant en paramètre un tableau de réels et retournant la moyenne des nombres du tableau. Testez la fonction !

Question 5.3 : Définir une fonction `minMax` prenant en paramètre un tableau de réels (et bien sûr son nombre d'éléments) et deux paramètres réels `min` et `max`. La fonction déterminera le minimum et le maximum des éléments du tableau et modifiera les paramètres `min` et `max` en conséquence. La fonction ne retournera aucune valeur.

Question 5.4 : Définir une fonction `saisieNotes` prenant en paramètre un tableau de réels (mais pas son nombre d'éléments). La fonction permettra à un utilisateur de saisir les notes d'un étudiant et de les stocker dans le tableau passé en paramètre. La saisie s'arrête lorsque l'utilisateur a saisi un nombre strictement inférieur à 0 ou strictement supérieur à 20. La fonction retournera le nombre de notes saisies par l'utilisateur.

Question 5.5 : Écrire le programme principal en utilisant les fonctions définies précédemment. Le tableau de notes sera déclaré comme un tableau de taille 1000 afin de s'assurer que l'utilisateur ne dépasse pas la capacité du tableau. Si l'utilisateur ne saisit aucune note, le programme devra afficher `Aucune note n'a été saisie pour l'étudiant` au lieu des minimum, maximum et moyenne. Ajouter également la mention de l'étudiant : redoublement (< 10), passable (entre 10 et 12), assez-bien (entre 12 et 14), bien (entre 14 et 16) et très bien (supérieur ou égal à 16).

TP9 : Tableaux et fonctions

Exercice 6 : Recherche dans des tableaux de caractères**

Question 6.1 : Écrire une fonction qui cherche si un élément appartient à un tableau de caractères (`car`). Le caractère recherché, le tableau et son nombre d'éléments seront les paramètres de la fonction.

Question 6.2 : Écrire une fonction qui compte le nombre d'occurrences d'un caractère dans un tableau. Le caractère recherché, le tableau et son nombre d'éléments seront les paramètres de la fonction.

Question 6.3 : Écrire une fonction qui prend deux tableaux en paramètres et qui teste si tous les éléments du premier tableau apparaissent au moins une fois dans le deuxième tableau. Il est possible d'utiliser dans le corps de cette fonction la fonction écrite pour la réponse à la question 6.1.

Exercice 7 : Tri très progressif d'un tableau**

Question 7.1 : Définir une fonction `initAlea` qui prend en paramètre un tableau d'entiers (et son nombre d'éléments) et qui initialise le tableau avec des nombres aléatoires compris entre 1 et 100.

Question 7.2 : Écrire un programme qui déclare un tableau de 25 entiers et qui l'initialise avec la fonction `initAlea`.

Question 7.3 : Écrire une fonction qui reçoit un tableau d'entiers et son nombre d'éléments, et qui renvoie vrai si le tableau est trié selon l'ordre croissant et faux sinon.

Question 7.4 : Complétez le programme principal (question 7.2) pour qu'il indique si le tableau initialisé aléatoirement est trié.

Question 7.5 : Écrire une fonction `inserer` qui prend en paramètre un tableau `tab` d'entiers et son nombre d'éléments. On suppose que tous les éléments de ce tableau `tab`, sauf le dernier, sont triés par ordre croissant. La fonction déplace le dernier élément dans le tableau de façon que l'ensemble des éléments soient triès (y compris le dernier !). Autrement dit, il s'agit d'insérer le dernier élément qui n'est pas encore trié à sa place.

Question 7.6 : Modifier le programme principal afin de trier le tableau au moyen de la fonction `inserer`.

Question 7.7 : Terminer le programme en déterminant si une même valeur apparait deux fois dans ce tableau trié. Lors de la conception de cet algorithme vous vous appuierez sur le fait que le tableau est désormais trié. Quelle conclusion en tirez vous sur la complexité de cette opération. Y-a-t-il moins ou plus de calcul quand le tableau est trié ?.

Exercice 8 : Tri alphabétique de tableaux de caractères**

Question 8.1 : Modifier la fonction `inserer` de l'Exercice 7 pour trier un tableau de prénoms par longueur des mots. Vous pourrez utiliser la fonction `int size(string s)` qui permet d'évaluer la taille d'une chaîne de chaînes de caractères ainsi que la fonction `strncpy(string copie, string s)` qui copie `s` dans `copy`.

Question 8.2 : Modifier la fonction `inserer` de l'Exercice 7 pour trier un tableau de prénoms par ordre alphabétique. Vous pourrez utiliser la fonction `strcmp(string s1, string s2)` qui retourne `0` si les chînes `s1` et `2` sont égales, un entier strictement négatif si `s1` est strictement inférieure à `s2` pour l'ordre alphabétique, un nombre strictement positif sinon.