Initiation à la Programmation C
Travaux Pratiques - L2
Utilisation de tableaux

Dans cette séance de travaux pratique, nous abordons les points suivants :

- l'utilisation de tableaux de N entiers.

Rappels :

La fonction `rand()` (de `<stdlib.h>`) renvoie un entier pseudo-aléatoire positif compris entre `0` et `RAND_MAX`.
La première valeur de la suite générée par des appels successifs à `rand` est fixée grâce à la fonction `srand(int seed);`. Cette première valeur de la suite (appelée graine, ou `seed` en anglais) définira toujours la même suite de valeurs.

Pour obtenir des valeurs entre `0` et `max - 1`, nous utiliserons alors l'instruction `rand() % max`.

Afin d'obtenir des valeurs différentes à chaque exécution, on initialise (une fois) la suite de valeurs avec une graine différente à chaque exécution. Une manière d'obtenir une graine différente à chaque exécution est d'utiliser la fonction `time()` de `<time.h>` qui renvoie le nombre de secondes écoulées depuis le 01 janvier 1970. Ceci s'exprime par l'instruction (à placer dans le `main`) :

```
srand(time(NULL));
```

▸ Exercice 1. Valeurs aléatoires

1. Écrire un programme qui affiche une suite de 20 valeurs pseudo-aléatoires, sans utiliser l'instruction `srand(time(NULL));`.

   Lancer plusieurs fois ce programme. puis ajouter l'instruction `srand(time(NULL));`.

   Expliquer pourquoi, en cours de développement, il peut être intéressant de ne pas modifier la graine à chaque exécution.

2. Écrire une fonction `void initAlea(int tab[], int taille, int max)` qui remplit les `taille` premières cases du tableau `tab` avec des entiers positifs aléatoires inférieurs à `max`.

▸ Exercice 2. Recherche d'un entier dans un tableau

Ecrire une fonction `int position(int tab[], int taille, int elt);` prenant en arguments un tableau, sa taille et un entier `elt`. Celle-ci renverra l'indice de la première occurrence de `elt` dans le tableau `tab` si `elt` apparaît dans les `taille` premiers éléments de `tab` et -1 sinon.

▸ Exercice 3. Somme de deux tableaux

On définit la *somme* `S = A + B` de deux tableaux `A` et `B` de même taille N comme étant le tableau `S` de N entiers définit par :

$$\forall i \in [\![0; N-1]\!],\ \texttt{S[i] = A[i] + B[i]}$$

Écrire une fonction `somme` recevant en arguments trois tableaux `A`, `B` et `S`, ainsi que leur taille commune. À la sortie de `somme`, le tableau `S` contiendra la somme des tableaux `A` et `B`.

▸ **Exercice 4. Somme de plusieurs tableaux**
On définit la *somme* `S = A0+ ···+ AM-1` de `M` tableaux de même taille N comme étant le tableau `S` de N entiers définit par :

$$\forall i \in [\![0; N-1]\!],\ \texttt{S[i] = A0[i] + ···+ AM-1[i]}$$

Modifier la fonction `somme` écrite précédemment pour qu'elle reçoive en arguments un tableau à deux dimensions à la place des tableaux `A` et `B`. On devra également donner en argument le nombre des tableaux sur lesquels on calcule la somme. Pour préciser la taille de la seconde dimension du tableau à deux dimensions dans les paramètres de la fonction, vous pourrez définir une constante globale `N` qui sera la taille maximale autorisée.

▸ **Exercice 5. Histogramme**
Écrire une fonction

```
void histogramme(int tab[][N], int nb_tab, int taille, int n_max, int histo[])
```

qui reçoit un tableau `tab` de `nb_tab` tableaux de `taille` entiers compris entre 0 et `n_max - 1`, ainsi qu'un tableau `histo` de `nb_tab * n_max + 1` entiers. Le tableau `histo` devra représenter, après l'appel de la fonction, l'histogramme de la somme `tab[0] + ... + tab[nb_tab - 1]` : pour tout `i`$\in [\![$`0;nb_tab * n_max`$]\!]$, `histo[i]` sera donc le nombre de valeurs égales à `i` dans le tableau `tab[0] + ···+ tab[nb_tab-1]`.

▸ **Exercice 6.** `Main`
Si ce n'est pas déjà fait, écrire une fonction `main()` utilisant les fonctions précédentes :

- remplissage aléatoire de trois tableaux de tailles `20` contenant des entiers entre `0` et `5` ;
- calcul et affiche l'histogramme de la somme de ces tableaux.