Séance 7&8 - Prise en main des tableaux statiques

A FAIRE SUR PAPIER

Exercice 1 Écrire un programme qui permet de créer un tableau `T` de 32 entiers tel que `T[i]=i`.

Exercice 2 Écrire un programme qui demande à l'utilisateur de saisir 10 entiers et qui les stocke dans un tableau `T`. Étendre votre programme pour qu'il affiche les éléments de `T` dans le sens inverse de leur saisie.

Exercice 3 Dans le programme ci-dessous, pour chacune des déclarations de tableau, donner sa taille ainsi que les valeurs des éléments :

```
int main(){
        int T1[4]={1,2,3,4};
        int T2[]={1,2,3,4,5,6};
        int T3[8]={1,2,3,4,5,6};
        int T4[2][2]={{1,2},{3,4}};
        int T5[][2]={{1},{2,3},{4}};
        return 0;
}
```

Exercice 4 Soient les déclarations suivantes :
`int A[8]={2,0,1,4,6,3,12,9};`
`int B[8];`
Écrire les programmes suivants :

1. Calcul de la somme préfixe de A dans B. La somme préfixe d'un tableau $T = [t_0, t_1, \cdots, t_{n-1}]$ est le tableau $[t_0, t_0 + t_1, t_0 + t_1 + t_2, \cdots, t_0 + t_1 + \cdots + t_{n-1}]$.
2. Placez dans le tableau B : tous les éléments pairs de A au début et les impairs de A à la fin.

Exercice 5 Écrire une fonction qui prend en paramètre un tableau de 20 réels et qui calcule la moyenne de ses éléments.

A FAIRE SUR MACHINE

Exercice 6 Écrire une fonction qui prend en paramètre un tableau de 10 entiers, un entier x et qui calcule le nombre d'occurrences de x dans le tableau. Écrire un programme permettant de tester votre fonction avec le tableau $\begin{bmatrix}2 & 3 & 4 & 2 & 3 & 1 & 8 & 2 & 5 & 9\end{bmatrix}$ et une valeurs saisie au clavier pour x. Vous testerez votre programme avec les valeurs de x suivantes : 2,3,4,5,6.

Exercice 7 Écrire une fonction qui prend en paramètre un entier x positif et un tableau Tx de 32 entiers, et qui calcule et stocke le codage binaire de x dans Tx (i.e. $x = \sum_{i=0}^{31} Tx[i] \times 2^i$). Écrire un programme qui demande à l'utilisateur de saisir un entier positif et qui affiche son codage binaire. Vous testerez avec les valeurs suivante :

- $123456 \Longrightarrow 0000\,0000\,0000\,0001\,1110\,0010\,0100\,0000$
- $1000000 \Longrightarrow 0000\,0000\,0000\,1111\,0100\,0010\,0100\,0000$
- $17 \Longrightarrow 0000\,0000\,0000\,0000\,0000\,0000\,0001\,0001$
- $2147483646 \Longrightarrow 0111\,1111\,1111\,1111\,1111\,1111\,1111\,1110$

Exercice 8 Écrire une fonction prenant en argument deux tableaux u et v représentant deux vecteurs de taille d et qui calcule leur produit scalaire. On rappelle que le produit scalaire de $\vec{u}$ et $\vec{v}$ est égal à :

$$\vec{u}.\vec{v} = \sum_{i=1}^{d} u_i \times v_i.$$

Tester cette fonction dans un programme avec les vecteurs :

- $\vec{u} = \begin{bmatrix}1 & 2 & 3 & 4\end{bmatrix}, \vec{v} = \begin{bmatrix}1 & 2 & 3 & 4\end{bmatrix}, \vec{u}.\vec{v} = 30$
- $\vec{u} = \begin{bmatrix}1 & 10 & 100 & 1000 & 10000\end{bmatrix}, \vec{v} = \begin{bmatrix}10000 & 1000 & 100 & 10 & 1\end{bmatrix}, \vec{u}.\vec{v} = 50000$

Exercice 9 Écrire un programme qui effectue une permutation des éléments d'un tableau de i (avec i signé induisant une permutation vers la droite si $i > 0$ et vers la gauche sinon). Par exemple si le tableau est $\{1, 2, 3, 4, 5, 6\}$ et $i = 2$, le résultat sera $\{5, 6, 1, 2, 3, 4\}$. Vous testerez votre programme avec les permutations $i = 0, 2, -3, -4$.

Exercice 10 Écrire une fonction qui ne renvoie aucune valeur et qui détermine la valeur maximale et la valeur minimale d'un tableau d'entiers (vous utiliserez les paramètre pour retourner les résultats). Écrire un programme principal permettant de tester cette fonction. Vous testerez avec le tableau $\{10, 11, 34, 2, 45, 49, 67\}$.

Exercice 11 Écrire une fonction d'affichage de tableau d'entiers. Elle servira à valider les exercices suivants. Vous testerez avec tous les tableaux des exercices précédents.

Exercice 12 Écrire une fonction prenant en argument un tableau d'entier T et qui le modifie de telle sorte que $T[i] = i * i$. Écrire un programme qui crée un tableau de 10 entiers initialisé avec votre fonction. Ce programme affichera le tableau après initialisation.

Exercice 13 Soit la suite (U_n) définie par $U_0 = 0$ et $U_{n+1} = U_n + 2n + 1$. Écrire une fonction qui initialise un tableau T avec les premiers termes de cette suite. Vous écrirez un programme permettant d'afficher un tel tableau avec les valeurs de n suivantes : 4, 10, 100 ;

Exercice 14 Écrire une fonction prenant en argument deux tableaux T_1 et T_2 et qui indique si les deux tableaux sont identiques. Vous testerez votre fonction avec les tableaux générés aux deux questions précédentes.

Exercice 15 Une chaîne de caractère est représentée par un tableau de caractère. En utilisant cela, définir les fonctions :

1. `prefixe` qui prend 2 chaînes de caractères `ch1` et `ch2` et qui teste si `ch1` est un préfixe de `ch2`.
2. `palindrome` qui prend 1 chaîne de caractères `ch1` et qui teste si `ch1` est un palindrome.

Ces deux fonctions doivent retourner un booléen.
Écrire les programmes associés permettant de tester vos fonctions. On initialisera les tableaux de caractères directement dans les programmes. On pourra utiliser l'initialisation simplifiée : `char mot[]="plateau";`

Exercice 16 (Pendu) Nous allons écrire un programme de jeu du pendu : Le but de ce jeu est de trouver un mot caché en devinant les lettres qui le compose. Vous avez droit à 5 erreurs. A chaque fois que vous énoncez une lettre n'appartenant pas au mot, vous accumulez une erreur. si votre lettre est présente dans le mot, toutes les occurrences de cette lettre sont dévoilées, et vous n'accumulez pas d'erreur. Si vous trouvez le mot avant d'avoir 5 erreurs vous avez gagné, sinon vous avez perdu.
Pour commencer, nous allons utiliser uniquement des mots de 8 lettres. Le mot à deviné sera écrit directement dans le programme principal.
Vous utiliserez 2 tableaux dans votre programme : un pour le mot à deviné, un pour le mot partiel que vous connaissez (le mot partiel pourra être initialisé avec le mot "- - - - - - - -").

— Écrire un fonction `afficheMot` qui prend un mot en paramètre (un tableau) et qui l'affiche à l'écran .
— Écrire le programme complet du jeu du pendu.

En incluant le fichier `<cstdlib>`, vous pourrez utilisez dans votre programme l'appel `system("clear");` qui permet de vider la fenêtre d'affichage.
Partie optionnelle : Comment faire pour que le programme puisse gérer des mots ayant une taille variable, tout en utilisant des tableaux statiques ? Modifiez votre programme pour qu'il gère des mots de tailles variables et qu'il demande de saisir le mot à deviner.