

Séance 7b: EXERCICES SUR LES TABLEAUX (PARTIE 2)

Université Paris Cité

Objectifs:

— Manipuler les tableaux

— Concevoir et programmer des algorithmes sur les tableaux

Dans cette séance, vous résoudrez des exercices et des problèmes sur des tableaux. Vous écrirez des boucles et définirez des fonctions intermédiaires pour rendre votre code plus lisible et concis.

Exercice 1 (Tableaux de mots,)

1. Écrire une procédure `letters2word` qui prend en argument un tableau de caractères `tab` et qui affiche le mot obtenu en concaténant ces lettres.

Contrat:

```
char[] tab={'p','l','a','c','a','r','d'}  
letters2word(tab) affiche "placard"
```

2. Écrire une procédure `stutterword` qui prend en argument un tableau de caractères `tab1` et un tableau d'entiers `tab2` et qui affiche le mot obtenu en concaténant les lettres du tableau `tab1`, comme suit : la lettre sur la position `i` dans `tab1` est répétée autant de fois que l'indique le numéro sur la position `i` dans `tab2`. La procédure doit afficher "Erreur" si les deux tableaux n'ont pas la même longueur.

Contrat:

```
char[] tab1={'a','b','c','d'}  
int[] tab2={2,2,3,4}  
stutterword(tab1,tab2) affiche "aabbcccdddd"
```

3. Écrire une fonction `word2letters` qui prend en argument un mot et renvoie le tableau de ses lettres.

Contrat:

```
word2letters("placard") renvoie {'p','l','a','c','a','r','d'}
```

4. Écrire une fonction `letters` qui prend en argument un mot (chaîne de caractères) `word` et qui renvoie le tableau de ses lettres, cette fois-ci sans doublons.

Conseil 1 : Il n'est pas possible de connaître à l'avance la taille du tableau résultat, ni à quels indices dans ce tableau les lettres seront ajoutées. Dans ce cas, on peut d'abord allouer un tableau trop grand, y écrire le résultat en gardant trace du nombre de lettres ajoutées puis, une fois le calcul terminé, recopier le résultat dans un nouveau tableau de la bonne taille.

Conseil 2 : Écrire d'abord une fonction `searchInPrefix(char[] tab, int n, char c)` qui renvoie `true` si le caractère `c` est égal à l'un des `n` premiers caractères du tableau `tab`, et `false` sinon.

Contrat:

letters("electroacoustique") renvoie
 {'e','l','c','t','r','o','a','u','s','i','q'}

□

Exercice 2 (Des ensembles, * – **)

1. Écrire une fonction search qui prend en argument un tableau d'entiers tab et un entier x et qui renvoie true si tab contient la valeur x, et false sinon.

Contrat:

int[] tab={6,20,12,1000,8}
 search(tab, 12) renvoie true
 search(tab, 50) renvoie false

2. Écrire une fonction union qui prend en argument deux tableaux d'entiers tab1, et tab2 (considérés sans doublons) et qui renvoie l'union de tab1 et tab2.

Contrat:

int[] tab1={6,20,12,1000,8}, tab2={2,8,6,7,12}
 union(tab1, tab2) renvoie {6,20,12,1000,8,2,7}

3. Écrire une fonction difference qui prend en argument deux tableaux d'entiers tab1, et tab2 (supposés sans doublons) et qui renvoie le tableau représentant la différence symétrique de tab1 et tab2. NB : La « différence symétrique » de A et B est l'ensemble des éléments qui appartiennent à A ou à B mais pas aux deux.

Contrat:

int[] tab1={6,20,12,1000,8}, tab2={2,8,6,7,12}
 difference(tab1, tab2) renvoie {20,1000,2,7}

□

Exercice 3 (Tri, ** – *)**

1. Écrire une fonction position qui prend en argument un tableau d'entiers tab, qu'on suppose déjà trié, et un entier x et qui renvoie la position dans tab dans laquelle on devrait insérer x, pour que le tableau obtenu reste trié.

Contrat:

int[] tab={0,2,4,6,7,8}
 position(tab,1) renvoie 1
 position(tab,-5) renvoie 0
 position(tab, 10) renvoie 6

2. Écrire une fonction insert qui prend en argument un tableau d'entiers tab, et deux entiers, pos et x, et qui renvoie le tableau obtenu en insérant l'élément x dans tab à la position pos. Si pos est strictement supérieur à la longueur de tab, la fonction renvoie le tableau sans modification.

Contrat:

int[] tab={2,5,4,3}
 insert(tab, 0, 1) renvoie {1,2,5,4,3}
 insert(tab, 2, 100) renvoie {2,5,100,4,3}
 insert(tab, 4, 42) renvoie {2,5,4,3,42}

3. Écrire une fonction sort qui prend en argument un tableau tab et qui renvoie le tableau trié (penser à utiliser les fonctions position et insert).

Contrat:

int[] tab = {40,1,20,3,8,6}
 sort(tab) renvoie {1,3,6,8,20,40}

□

Exercice 4 (Circulaire, *)**

Écrire une fonction `circulaire` qui prend en argument deux tableaux d'entiers `tab1` et `tab2` et qui renvoie `true` si `tab2` est une permutation circulaire de `tab1`, et `false` sinon (en particulier si les tableaux n'ont pas la même longueur).

Conseil : utiliser la fonction `shift` de l'exercice 4 du TP6b.

Contrat:

```
int[] tab1={1,2,3,4,5}
int[] tab2={3,4,5,1,2}
int[] tab3={3,5,4,1,2}
circulaire(tab1,tab2) renvoie true
circulaire(tab1, tab3) renvoie false
```

□