Yann.Secq@univ-lille.fr
Fabien.Delecroix@univ-lille.fr

TD n°10

Objectifs: Comprendre les deux modes de passage des paramètres: par valeur et par référence. Retravailler l'ensemble des notions.

Exercice 1 : Passage des paramètres par valeur et par référence

Dans cet exercice, nous allons revenir sur les questions de passage par valeur (copie) ou par référence (adresse). Les types simples (ou primitifs), c'est-à-dire `byte`, `short`, `int`, `long`, `float`, `double`, `char` et `boolean` sont passés par valeur, c'est-à-dire que lors de l'appel d'une fonction, une copie de la valeur est transmise à la fonction. Par contre, les types plus complexes comme `String` ou les tableaux (`int[]`, `String[][]` ...) sont passés par référence, c'est-à-dire que lors de l'appel d'une fonction, l'adresse en mémoire de la variable est transmise à la fonction. Dit autrement, l'appelant (celui qui appelle la fonction) et l'appelé (la fonction qui est exécutée) partagent la même variable. Les modifications faites au sein de la fonction appelée seront effectives aussi sur la variable de l'appelant.

Pour chacun des corps de programme qui suivent, indiquez ce qu'il va afficher et expliquer votre réponse.

Corps du Programme n°1

```
1  void inc(int i) {
2    i = i+1;
3  }
4  void algorithm() {
5    int age = 99;
6    inc(age);
7    print(age);
8  }
```

Corps du Programme n°2

```
1  int inc(int i) {
2    return i+1;
3  }
4  void algorithm() {
5    int age = 99;
6    age = inc(age);
7    print(age);
8  }
```

Corps du Programme n°3

```
1  void initialize(int[] t) {
2    t = new int[] {1, 2, 3, 4, 5};
3  }
4  void algorithm() {
5    int[] serie = new int[5];
6    initialize(serie);
7    for (int i=0; i<length(serie); i++) {
8      print(serie[i]);
9    }
10 }
```

Corps du Programme n°4

```
1  int[] initialize() {
2    return new int[] {1, 2, 3, 4, 5};
3  }
4  void algorithm() {
5    int[] serie = initialize();
6    for (int i=0; i<length(serie); i++) {
7      print(serie[i]);
8    }
9  }
```

Corps du Programme n°5

```
1  void initialize(int[] t) {
2    for (int i=0; i<length(t); i++) {
3      t[i] = i+1;
4    }
5  }
6  void algorithm() {
7    int[] serie = new int[5];
8    initialize(serie);
9    for (int i=0; i<length(serie); i++) {
10     print(serie[i]);
11   }
12 }
```

Exercice 2 : Utiliser le passage par référence des tableaux

L'objectif principal de cet exercice est de s'exercer au mode de passage des paramètres par référence. Vous devez écrire un programme qui va, 10 fois de suite, générer un mot de passe aléatoirement puis l'afficher. On considère qu'un mot de passe est une suite de 14 caractères compris dans l'intervalle $[33; 126]$ de la table ASCII (càd qui peut contenir

des lettres majuscules et minuscules, des chiffres et des symboles de ponctuation). Nous allons représenter un mot de passe par un tableau de caractères qui sera rempli aléatoirement à chaque tour de boucle.

Nous allons décomposer le problème pour avoir l'algorithme principal suivant:

```
1 void algorithm() {
2     // Allouer un tableau qui va contenir les mots de passe
3     char[] mdp = new char[14];
4     // Remplir et afficher 10 mots de passe
5     for (int i = 0; i < 10; i++) {
6         remplissageAlea(mdp);
7         afficher(mdp);
8     }
9 }
```

1. Écrire la fonction `void afficher(char[] tab)` qui affiche les caractères contenus dans le tableau de caractères donnée en paramètre.
2. Écrire la fonction `void remplissageAlea(char[] tab)` qui remplit le tableau donné en paramètre avec des caractères aléatoires dans l'intervalle $[33; 126]$. On suppose que le tableau `tab` est déjà alloué. Notez que la taille du tableau détermine la longueur du mot de passe à générer.

Exercice 3 : Valeurs voisines dans un tableau

On souhaite dans cet exercice réaliser une fonction qui détecte si deux valeurs sont voisines dans un tableau de chaînes de caractères.

1. Proposez une signature pour votre fonction (type de retour, nom, type et nom de chaque paramètre)
2. Avant d'écrire cette fonction, afin de mieux considérer différents cas de figures, commençons par écrire la fonction qui la testera. Dans celle-ci, vous devrez d'abord définir 2 tableaux : `tab1` contenant les valeurs `"Clément"`, `"Baptiste"`, `"Anaïs"`, `"Dalila"`, `"Clément"` dans cet ordre et `tab2` contenant uniquement la valeur `"Han"`. Écrivez ensuite le code permettant de vérifier les scénarios suivants.

Scénario	Paramètres			Résultat attendu
n°1	tab1	"Dalila"	"Clément"	true
n°2	tab1	"Anaïs"	"Baptiste"	true
n°3	tab1	"Anaïs"	"Clément"	false
n°4	tab2	"Han"	"Han"	false

3. Écrivez maintenant le code de votre fonction permettant de savoir si deux chaînes de caractères sont voisines dans un tableau.
4. Prolongement Écrivez une fonction similaire et des tests pertinents mais cette fois pour un tableau à 2 dimensions en considérant le voisinage dans les 4 directions.