Langage et Programmation – 5ème séance

Fabien Tarissan

1 Les tableaux

Exercice 1 — *Premières opérations*

- Créer un tableau `t` contenant les valeurs | 1 | 3 | 5 | 7 | 9 | de deux manières : directement, et avec l'utilisation de range.
- Afficher la taille de `t`
- Écrire une fonction `afficheTab` permettant l'affichage des éléments d'un tableau passé en argument.
- Écrire un programme demandant à l'utilisateur un entier n et créant un tableau `tab0n` contenant les valeurs de 0 à n (inclus).
- Écrire un programme demandant à l'utilisateur la taille du tableau voulue puis demandant ensuite à l'utilisateur les valeurs à mémoriser dans le tableau. [1]
- Écrire une fonction qui, étant donnés un tableau d'entiers, et deux indices `i` et `j`, échange les valeurs de ces deux indices si c'est possible, ne fait rien sinon (*i.e.* si les indices ne correspondent pas à des cases du tableau). Comparer cette fonction avec la fonction `echange` de la feuille de la semaine dernière. Pourquoi celle-ci fonctionne ?

Exercice 2 — *Moyenne* Écrire une fonction `moyenne` qui calcule la moyenne des éléments d'un tableau d'entiers.

> *Variante plus simple : calculer la somme des éléments d'un tableau.*
> *Exercice complémentaire : calculer l'écart-type.*

Exercice 3 — *Min et max* Soit un tableau d'entiers naturels non vide. Écrire une fonction `maxTableau` qui calcule le maximum contenu dans le tableau. Donner la variante calculant le minimum.

> *En complément, calculer l'écart maximum dans un tableau. Quelle est la complexité de cette fonction ? Comment améliorer ça en utilisant un couple ?*

Exercice 4 — *Appartenance* Sans utiliser la forme `in`, écrire une fonction `appartient` qui teste l'appartenance d'un entier dans un tableau.

Exercice 5 — *Occurence* Écrire une fonction `nbOccurence` qui compte le nombre d'occurence d'un entier dans un tableau.

> *En complément (mais plus dur) : calculer la distribution des valeurs du tableau, c'est-à-dire, pour chaque valeur v du tableau, le nombre d'occurence de v dans le tableau.*

Exercice 6 — *Occurence du maximum* Écrire une fonction `nbOccurenceMax` qui compte le nombre d'occurence du maximum dans un tableau. Quelle est la complexité de votre fonction. Peut-on faire mieux ?

Exercice 7 — *Égalité* Qu'affiche le code suivant :

1. il sera utile d'encapsuler primitive dans une fonction `readTableauInt(n)` afin de la réutiliser par la suite.

```
t1=[1,2,3]
t2=[1,2,3]
t1 is t2
t1 == t2
```

Expliquer pourquoi. Proposer une fonction qui teste l'égalité de deux tableaux d'entiers.

Mise en évidence de la sémantique de l'opérateur is. C'est l'égalité dite physique. *Alors que l'opérateur == teste l'égalité* structurelle

Exercice 8 — *Même contenu* Écrire une fonction qui teste si deux tableaux d'entiers de même taille contiennent les mêmes valeurs. On supposera dans un premier temps que tous les éléments d'un tableau sont distincts. Que pensez-vous de la complexité de votre fonction. Peut-on faire mieux ?

Refléchir à la variante dans laquelle les éléments peuvent être répétés.

Cet exercice est en lien avec la partie algorithmique. On peut faire mieux si on connaît les algorithmes de tris ($O(nlog(n))$), ce qui permet également de répondre efficacement à la variante autorisant la duplication des éléments.

2 Les classes

Exercice 9 — *Vecteurs* Réaliser une classe `Vecteur` permettant de manipuler des vecteurs à trois composantes et disposant :

1. D'un constructeur à trois arguments.
2. D'une méthode d'affichage des coordonnées du vecteur.
3. D'une méthode qui teste si un vecteur est le vecteur nul.
4. D'une méthode calculant la norme euclidienne d'un vecteur :
$$||(x, y, z)|| = \sqrt{x^2 + y^2 + z^2}$$
5. D'une méthode calculant la somme de deux vecteurs.
6. D'une méthode calculant le produit scalaire de deux vecteurs :
$$\langle (x, y, z), (x', y', z') \rangle = xx' + yy' + zz'$$
7. D'une méthode calculant le produit vectoriel de deux vecteurs.
$$(x, y, z) \wedge (x', y', z') = (yz' - y'z, zx' - xz', xy' - x'y)$$
8. D'une méthode testant si deux vecteurs sont co-linéaires.
9. D'une méthode testant si deux vecteurs sont orthogonaux.

Exercice 10 — *Cave* Afin de stocker du liquide, on est amené à définir le concept de bouteille. Une bouteille a une contenance maximale et une contenance réelle (sa contenance à un instant donné). Lors de la création d'une bouteille son nom et sa capacité maximale est donnée, ainsi que si elle ferme à vis ou non. La bouteille est pleine au début.

On veut pouvoir :
– remplir partiellement la bouteille d'une certaine quantité,
– la remplir complètement,
– la vider d'une certaine quantité,
– la vider complètement,
– connaître sa contenance,
– afficher ses caractéristiques (contenance, quantité actuelle, fermeture à vis ou non).

Bien évidemment, on ne peut pas remplir plus que la contenance de la bouteille, ni vider d'une quantité supérieure à la quantité actuelle. En cas de remplissage avec une quantité plus grande que ne peut contenir la bouteille, celle-ci sera remplie à son maximum possible.

Proposez une implémentation de la classe `Bouteille` intégrant les contraintes ci-dessus.

À l'aide de cette classe, proposez une classe `Cave` qui permette de gérer une cave.

3 Des exercices en vrac

Exercice 11 — *D'autres exercices* Sans liens les uns avec les autres mais groupés par intérêt :

1. Écrire une fonction qui calcule la distance parcourue en fonction de la vitesse et du temps.
2. Écrire une fonction qui calcule la période d'un pendule en fonction de sa longueur.
3. Écrire une fonction qui renvoie la valeur absolue d'un nombre réel.
4. Écrire une fonction qui calcule la valeur d'un polynôme du second degré en x (4 paramètres)
5. Écrire une fonction calcule les racines d'un polynômee du second degré (3 paramètres)

   Sur les entiers naturels :
6. Écrire une fonction qui donne la liste des diviseurs d'un entiers
7. Écrire une fonction qui décompose un nombre en facteur de nombres premiers
8. Écrire une fonction qui affiche les n premiers nombres premiers
9. Écrire une fonction qui affiche les entiers naturels inférieurs à un entier n en utilisant le crible d'Eratosthène. Comparer son efficacité avec la solution qui découle de l'exercice 7 de la séance 3.

   Sur les tableaux :
10. Écrire une fonction qui calcule l'écart-type d'un tableau de nombres.
11. Écrire une fonction qui calcule l'écart maximum dans tableau de nombres.
12. Écrire une fonction qui teste si un tableau de nombre est croissant.
13. Écrire une fonction qui calcule la distribution des valeurs d'un tableau de nombres.
14. Écrire une fonction qui inverse l'ordre des éléments d'un tableau : variante qui fait la modification *en place* (c'est à dire change les valeurs du tableau donné en paramètre) et variante qui crée un nouveau tableau.

    Sur la récursion :
15. Écrire une fonction qui calcule la somme des n premiers nombres pairs.
16. Écrire une fonction qui calcule une approximation de π, de $cos(x)$: cf exercice 5 de la séance 4
17. Écrire une fonction qui calcule le coefficient binomial

$$C_p^n = \begin{cases} 1 & \text{si } p = 0 \\ 1 & \text{si } n = p \\ C_{p-1}^{n-1} + C_p^{n-1} & \text{sinon} \end{cases}$$