Tableaux

Temps d'étude conseillé : 4h

Po est un Kakapo [1] triste. Son ami, Jo l'ibijau [2], lui a envoyé des photos de lui alors qu'il était sur la plage de Copacabana en train de se dorer le plumage. Po se prend de temps en temps à rêver au *summer body* de l'ibijau, et souhaiterait de tout sons cœur avoir le même. Il décide donc de l'appeler afin d'obtenir des conseils.

1 Introduction : Po surveille sa ligne

« Jo, tu as vraiment un corps de rêve et un plumage si parfait. Peux-tu m'expliquer comment tu fais ? »
« Ton problème est que tu es trop gourmand, Po » lui répond alors Jo.
« Tu devrais peut-être surveiller la quantité de nourriture que tu manges chaque jour, vu que le sport n'est pas ton fort. On analysera ça ensemble dans quelques semaines. »

Po décide alors de se prendre en main et d'écouter les conseils de son ami. Il souhaiterait coder un programme lui permettant de calculer le poids de ses rations journalières. En l'état actuel de ses connaissances informatiques, Po est incapable de réaliser un tel programme. Mais Jo, qui a suivi les cours de l'ESIEA il y a quelques années, lui explique qu'un tel programme est réalisable à l'aide d'un *tableau*. Il lui explique alors la chose suivante.

« Il est possible de simuler l'utilisation d'une quantité arbitraire de variables au travers d'un *tableau*. Il s'agit d'un **ensemble de valeurs**, stockées de façon **contigües en mémoire**, et **toutes du même type**. Chacune de ces valeurs se trouve dans une *case* repérée à l'aide d'un *indice*. De façon bizarre, mais explicable, la première case est numérotée 0. Voilà un exemple de tableau contenant cinq nombres flottants.

Indice	0	1	2	3	4
Valeur	13.37	42	-7.3	0	737.37

Par exemple, grâce à un tableau, tu pourras stocker 15 entiers, et chacun de ces entiers correspondra au poids de nourriture que tu ingurgites en une journée. Tu auras ainsi surveillé ton régime alimentaire durant 15 jours, qui sera très simple à analyser par la suite. »

Po lui répond alors : « C'est parfait Jo, mais je ne vois pas bien en quoi ce tableau change la donne par rapport à 15 variables que j'aurais déclarées et affectées à l'aide de 15 appels à la fonction `scanf()` ? »

« Et bien si tu veux réutiliser ton programme une autre fois, répondit l'ibijau, tu devras saisir à nouveau exactement 15 valeurs, soit 15 jours. Qui te dit d'ici là qu'il ne faudra pas surveiller ton poids plus longtemps ? Et puis grâce au tableau, tu auras le gros

1. `https://youtu.be/IBIDyINTDWw`, attention à ne pas vous perdre sur Internet !
2. `https://youtu.be/b59ZM7fXOSs`, ça y est, vous vous êtes perdu...

avantage de pouvoir condenser ton code en utilisant une simple instruction de boucle pour accéder à tes données ! Aies confiance, ce n'est pas si compliqué. »

Encore plein de doutes, Po se résigna donc à écouter l'ibijau et à réaliser les exercices qu'il lui demandait de pratiquer afin d'oublier sa triste image de lui-même.

« Comme je t'ai dit, l'ensemble des valeurs d'un tableau est accessible à l'aide du nom du tableau et de l'indice de la case à laquelle tu veux accéder. Mais pour cela, il te faut déclarer un tableau. Tu peux par exemple utiliser l'instruction suivante. »

```
int tab[10];  // Déclaration d'un tableau de 10 entiers
```

« Cette instruction a donc réservé **10** emplacements dans la mémoire, indicés de 0 à 9, chaque emplacement permettant de stocker un entier à l'aide du mot clé **int**. Tu pourras par la suite accéder à tes éléments à l'aide du nom du tableau, qui est ici **tab**. Note bien que le nombre d'éléments de ton tableau (10 ici) est fixe. **Cette taille doit être une constante** !

« Il est alors possible de remplir les premières cases de ton tableau en les affectant lors de la déclaration. Écris les valeurs entre des accolades séparées par des virgules de la façon suivante. La première case (d'indice 0) aura la valeur 3, la seconde la valeur -2 etc. Tu peux attribuer une valeurs aux premières cases, et si il manque des éléments, les cases suivantes seront à 0. »

```
short T[7] = {3, -2, 1};  // Declaration d'un tableau de 7 short
```

« Pour bien comprendre, voici à quoi ressemble le tableau `T` que je viens de définir. »

Indice	0	1	2	3	4	5	6
Valeur	3	-2	1	0	0	0	0

TD **Exercice 1.**

2 Application : Po se surveille

Ragaillardi par cette introduction, Po demande plus de précisions à Jo sur le fonctionnement d'un tableau. Il lui demande alors comment il peut accéder à l'élément situé dans une case particulière du tableau. L'ibijau répond alors : « Comme je te l'ai dit tout à l'heure, tu as simplement besoin du nom de ton tableau et de l'indice de la case à laquelle tu veux accéder. Tu pourras alors lire la valeur présente dans cette case et même la modifier. Recopies l'exemple suivant pour tout comprendre, puis fais mon petit exercice pour bien comprendre le fonctionnement de l'accès au tableau. »

```
int i = 0;
short T[7] = {3, -2, 1};
T[0] = 14;    // Modification de la case 0 du tableau T
T[3] = T[i];  // Modification de la case 3 du tableau T
for (i=0; i<7; i++)
  T[i] = i;  // Parcours et remplissage du tableau
```

TD **Exercice 2.**

« Ca y est ! Je sais comment faire ! Tu vas enfin pouvoir analyser ma consommation de fruits ! » s'écria Po. Jo lui répondit alors que si il savait créer son programme, il n'allait certainement pas être très modulable, car il ne comporterait aucun appel de fonction manipulant un tableau. En effet, Po a omis ce précepte important, et demanda donc à l'ibijau plus de précisions à ce sujet.

« Supposons que tu veuilles coder la fonction `func()`, qui prend en paramètres un tableau, et qui renvoie la moyenne des `n` premiers éléments présents dans ce tableau. Tu vas devoir dans un premier temps rédiger le prototype de la fonction de la façon suivante. »

```
float func (int tab[], int n)
```

Jo lui expliqua alors que pour passer un tableau à une fonction en tant que paramètres, il était obligatoire d'écrire le type de données stockées dans le tableau, suivi du nom du tableau et des crochets. Nul besoin de mettre la taille du tableau entre crochets, cette dernière est passée également en argument de la fonction. Il suffit alors de réaliser une boucle sur les `n` premiers éléments du tableau afin de les sommer dans une variable `resultat`, puis de diviser cette variable par `n` avant de la renvoyer pour obtenir le résultat souhaité.

« Oh mon kakapo, tu es vraiment trop fort ! C'est un parfait exemple de modularité que tu viens de me montrer. Mais comment puis-je appeler cette fonction sur mon tableau `tab` qui contient mes `n` données ? »

Jo lui présenta alors l'instruction suivante montrant qu'il suffisait de mettre le nom du tableau dans l'appel de fonction (pas besoin de crochets ou autres) et Po réalisa les exercices donnés par l'ibijau pour s'entrainer.

```
int tab[10];
float res;
res = func (tab, 4);
```

TD Exercice 3.

Encore un peu timide, et affamé par autant de réflexion, Po demanda encore quelques précisions à Jo sur la modification d'un tableau. Il lui assura que lorsqu'un tableau est passé en argument d'une fonction, il est possible de modifier les valeurs de ce tableau dans la fonction, de sorte que ces modifications subsistent encore après la fin de la fonction.

« Lorsque tu réalises l'appel d'une fonction dont le prototype est `void f(int tab[])` sur le tableau `T`, tu écris `f(T)`. En réalité, `T` est considéré comme un pointeur contenant l'adresse de la première case du tableau `T`. Ainsi, lorsque tu écris dans ta fonction l'instruction `tab[i] = x`, tu modifies ce qu'il y a `i` cases après l'adresse stockée dans `tab`, ce qui revient à modifier la case `i` du tableau `T`. Tout cela semble très compliqué, mais ne t'inquiètes pas, ton maître saura t'expliquer cela très prochainement. Retiens simplement qu'il est possible de modifier un tableau en le passant à une fonction. »

« Courage mon poto Po, dit le potoo Jo[3], faut pas se laisser aller ! »

TD Exercice 4.

3. Répétez cette phrase cinq fois très rapidement sans vous tromper.

3 Tableaux 2D : Po a faim

Au cours d'un voyage d'affaire, Jo rendit visite à Po à l'issue des quinze jours.

« J'ai un souci avec ton tableau, Po. Tu inscrits des valeurs qui ont l'air complètement fantaisistes. Tu manges vraiment 2kg de nourriture par jour alors que tu ne pèses que 4kg ? »

« Mais j'ai faim ! Et j'ai perdu le vrai tableau. Celui-ci ne contient que des valeurs aléatoires. » lui avoua Po.

« Dans ce cas, nous allons adapter ton régime alimentaire. Puisque tu es difficile question nourriture, ce sera facile. Ton régime alimentaire est réduit à 10 aliments différents (dont le gravier, je n'ai jamais compris pourquoi). Je vais donner des points à ces aliments. Tu vas devoir créer un planning sur deux semaines donnant les aliments et leur quantité à manger chaque jour. Ton objectif sera de ne pas dépasser 100 points par jour. Je te laisse, j'ai un shooting photo pour Potoos Illustrated. »

Abattu par ses informations et sa mauvaise blague, Po retourna dépité dans sa forêt et essaya de grimper sur une branche. Incapable de l'atteindre, et énervé par sa condition, il appela à nouveau Jo pour se faire aider.

« Il existe le concept des *tableaux à deux dimensions*. Il s'agit en quelque sorte de tableaux de tableaux, qui permettent de stocker un ensemble de séries de données. Tu pourrais ainsi avoir un tableau de plusieurs lignes, et pour chaque ligne, dix cases indiquant la quantité de chaque aliment à manger dans la journée. »

Jo l'ibijau expliqua alors qu'un tableau 2D se déclarait un peu de la même façon qu'un tableau 1D, en écrivant d'abord le type des données, le nom du tableau puis la taille de chaque dimension entre crochets successifs. Et parce qu'il savait qu'un exemple était ce qu'il y avait de mieux pour Po, il lui montra la série d'exemples suivants.

```
int tab2D[3][2];   // Déclaration d'un tableau 3 lignes, 2 colonnes
short matrice[2][3] = {{1,2,3},{4,5,6}};
matrice[0][1] = 2; // Accès et modification d'un élément
```

« Comme tu peux le voir, expliqua Jo, là encore, il est possible de remplir un tableau 2D lors de sa déclaration. Le tableau nommé `matrice` comporte par exemple deux lignes de trois éléments et peut être schématisé de la façon suivante. »

	col.0	col.1	col.2
ligne 0	1	2	3
ligne 1	4	5	6

« Pour accéder à l'élément de la ligne 0, colonne 1, tu écris donc `matrice[0][1]`, qui vaut 2. En effet, `matrice[0]` est considéré comme un tableau, non comme une valeur. Il est donc impossible d'écrire `matrice[0] = 10`. En revanche, puisque `matrice[0]` est un tableau à 3 éléments, on peut accéder à l'élément d'indice `1` en écrivant `matrice[0][1]`. »

« Il est ainsi possible, comme pour les tableaux 1D, de parcourir l'ensemble des valeurs de ton tableau 2D. Pour cela, il te faut parcourir l'ensemble des lignes, appelées ici `matrice[i]`, pour `i` allant de 0 à 1, puis pour chaque ligne, parcourir l'ensemble des

valeurs `matrice[i][j]` pour `j` allant de 0 à 2. Note que l'on appelle aussi les tableaux 2D des matrices. Entraine-toi sur les quelques exemples suivants pour bien comprendre. »

TD **Exercice 5.**

« Et comme tout à l'heure, demanda Po, est-il possible d'utiliser des fonctions avec des matrices ? » Jo lui répondu par l'affirmative en lui présentant le prototype de la fonction suivante.

```
float Moyenne (int mat[][2], int nbLignes, int nbCol);
```

Il lui expliqua que pour passer une matrice en paramètre d'une fonction, il était obligatoire d'indiquer le nombre d'éléments présents sur chaque ligne en renseignant ce nombre dans les seconds crochets. En effet, si l'on n'indiquait pas cette valeur, comment savoir où se situent les données en mémoire de la ligne 1, si l'on ne sait pas de combien d'éléments on doit se décaler pour y accéder ? Il indiqua également qu'il était souvent intéressant de donner également le nombre de lignes et de colonnes afin de se déplacer sans soucis dans la matrice. Finalement, l'appel de la fonction était ce qu'il y avait de plus simple à réaliser car comme pour les tableaux 1D, il suffisait de donner le nom du tableau comme dans l'exemple ci-dessous.

```
int tab2D[3][2];
float res;
res = Moyenne (tab2D, 3, 2);
```

« Tu peux donc réaliser ton programme, indiqua Jo. Nous allons d'abord créer un premier tableau qui contiendra le nombre de points de chaque aliment. Ensuite, tu devras coder une fonction permettant de remplir une matrice détaillant le nombre d'aliments que tu pourras manger chaque jour. La somme des points calculée sur chaque ligne ne devra pas dépasser 100. Voici le tableau des points par aliment. »

Aliment	Points
Baie de Rimu	7
Sève de Rimu	3
Feuilles de Rimu	2
Branches de Rimu	1
Amandes	8
Noix de Macadamia	10
Pollen	2
Gravier	4
Terre	7
Cactus	3

Po se sentit bien découragé face à l'ampleur de cette tâche. Mais heureusement, l'ibijau était un bon professeur et lui proposa de bons exercices pour s'entrainer.

TD **Exercice 6.**

4 Conclusion

Suite à la mise en place de ce régime draconien, Po a réussi à perdre quelques grammes. Il croise à nouveau son ami Jo l'ibijau qui vérifie sa nouvelle corpulence dans son nouveau cabinet de diététicien pour Strigopinae.

« Salut, Po. Je vais te demander de te mettre à nu pour vérifier ton poids. Tu as un déplumoir sur ta droite. »

Jo retira alors ses plumes superflues et monta sur la balance.

« Eh bien, mon poto Po, félicitations ! Sans atteindre mon corps de rève, on peut dire que tu es maintenant le plus affuté des kakapos ! Tes combats d'arène se passeront nettement mieux maintenant ! »

« Oh merci Jo, tu fais de moi un nouvel oiseau ! Je n'ai enfin plus besoin de me cacher derrière des fougères et je vais enfin pouvoir voler de mes propres ailes ! »

Sur ces mots, Po ne se sent plus de joie. Il ouvre grand ses ailes et saute par la fenêtre.

Trois mois après son accident [4], Po récupéra enfin sa place de roi des kakapo dans sa tribu de l'île de Maud.

Jo quant à lui devint un diététicien renommé. Il développa un programme en C afin d'aider tous les kakapos du monde à perdre du poids. Suite à ce succès, l'ibijau succomba finalement à la facilité du Python et perdit ses amis qui ne le reconnaissaient plus.

FIN

4. Ne sachant pas voler, Po s'est foulé une patte en retombant de la fenêtre du rez-de-chaussée du cabinet de Jo.