Feuille de révisions n°4 - Programmation en C : manipulation de tableaux

1 Création de tableaux

R. 4-1 Écrire une fonction prenant en argument un entier naturel non nul n et retournant un tableau de n booléens rempli de `false` à l'aide d'une boucle `while`.

R. 4-2 Écrire une fonction prenant en argument un entier naturel non nul n et retournant un tableau contenant les entiers de 1 à n à l'aide d'une boucle `for`.

R. 4-3 Écrire un **programme** qui crée le tableau contenant les entiers 1, 2, 4, 8, 12, 56, -3 et 19, puis qui affiche un à un ces éléments.

R. 4-4 Écrire une fonction prenant en argument un entier naturel non nul n, et deux entiers relatifs a et b tels que $a < b$ et et retournant un tableau de n entiers aléatoires tirés uniformément dans $[\![a, b[\![$. On veillera à ce que le tableau construit par l'appel à cette fonction change potentiellement à chaque exécution.

2 Agrégation

R. 4-5 Écrire une fonction testant l'existence d'un `0` dans un tableau d'entiers parcouru par boucle `while`.

R. 4-6 Écrire une fonction calculant l'indice du premier `0` dans un tableau d'entiers parcouru par boucle `while` arrêtée dès que possible, sans utiliser de `break`, ni de `return` dans la boucle. La fonction devra retourner `-1` si le tableau ne contient aucun `0`.

R. 4-7 Écrire une fonction calculant l'indice du premier `0` dans un tableau d'entiers parcouru par boucle `while` arrêtée dès que possible grâce à une instruction `return` dans la boucle. La fonction devra retourner `-1` si le tableau ne contient aucun `0`.

R. 4-8 Écrire une fonction calculant l'indice du premier `0` dans un tableau d'entiers parcouru par boucle `for` arrêtée dès que possible. La fonction devra retourner `-1` si le tableau ne contient aucun `0`.

R. 4-9 Écrire une fonction calculant l'indice du dernier `0` dans un tableau d'entiers parcouru par boucle `while` arrêtée dès que possible grâce à une instruction `return` dans la boucle. La fonction devra retourner `-1` si le tableau ne contient aucun `0`.

R. 4-10 Écrire une fonction prenant en argument un tableau d'entiers et calculant la somme des éléments du tableau. On fournira une implémentation au moyen d'une boucle `for`.

R. 4-11 Écrire une fonction calculant l'indice du minimum dans un tableau d'entiers non vide parcouru par une boucle `for`.

R. 4-12 Écrire une fonction calculant la conjonction d'un tableau de booléens parcouru par une boucle `while`.

R. 4-13 Écrire une fonction prenant en argument un tableau d'entiers à valeurs dans $[\![0, m-1]\!]$ où m est donné en argument, et calculant l'entier de $[\![0, m-1]\!]$ ayant le plus d'occurrences dans le tableau. On fournira une implémentation en $\mathcal{O}(n+m)$.

3 Tableaux à partir d'un tableau

R. 4-14 Écrire une fonction qui copie un tableau.

R. 4-15 Écrire une fonction prenant en argument un tableau `t` de taille n et une valeur par défaut `x` de même type que celui des éléments du tableau et retournant un tableau de taille $2n$ dont les n premiers éléments sont ceux de `t` et les n derniers contiennent la valeur par défaut.

R. 4-16 Écrire une fonction prenant en argument un tableau d'entiers `t` et retournant le tableau miroir de `t`.

R. 4-17 Écrire une fonction prenant en argument un tableau d'entiers `tailles` et retournant `t` un tableau de tableaux d'entiers initialisé à 0, de même taille que le tableau `tailles`, et tel que pour tout indice `i` valide pour ces tableaux, `t[i]` est de taille `tailles[i]`.

R. 4-18 Écrire une fonction prenant en argument un tableau d'entiers `t` et retournant le tableau des sommes cumulées du tableau `t`. Ainsi la case d'indice `i` du tableau résultat doit contenir `t[0] + t[1] + ... + t[i]`. On fournira une implémentation de complexité linéaire en la taille du tableau.

R. 4-19 Écrire une fonction qui prend en arguments deux tableaux d'entiers triés et qui crée le tableau trié obtenu en interclassant les éléments des deux tableaux. Le nombre d'occurrence de chaque élément dans le tableau résultat est la somme de ceux dans chacun des tableaux en argument. Cette fonction devra être en complexité linéaire, c'est-à-dire en $\mathcal{O}(n_1 + n_2)$ où n_1 et n_2 sont respectivement les tailles des deux tableaux pris en argument.

R. 4-20 Écrire une fonction permettant de distinguer les sous-séquences (non vides) croissantes maximales pour l'extension à gauche d'un tableau d'entiers non vide en indiquant où elles commencent, c'est-à-dire retournant le tableau trié des indices où commencent ces sous-séquences, muni de sa taille, *i.e.* le nombre de telles sous-séquences. Par exemple pour `{1, 4, 5, 8, 7, 5, 2, 3, 4, 9, 3, 3, 3, 5, 5, 7, 2}`, le tableau des indices de découpage est `{0, 4, 5, 6, 10, 16}` correspondant aux 6 sous-séquences `{1, 4, 5, 8}`, `{7}`, `{5}`, `{2, 3, 4, 9}`, `{3, 3, 3, 5, 5, 7}` et `{2}`. Cette fonction pourra allouer un tableau temporaire, mais ne devra parcourir qu'une fois de tableau pris en argument.

R. 4-21 Écrire une fonction permettant de distinguer les sous-séquences (non vides) croissantes maximales pour l'extension à gauche d'un tableau d'entiers non vide en indiquant où elles commencent, c'est-à-dire retournant le tableau trié des indices où commencent ces sous-séquences, muni de sa taille, *i.e.* le nombre de telles sous-séquences. Par exemple pour `{1, 4, 5, 8, 7, 5, 2, 3, 4, 9, 3, 3, 3, 5, 5, 7, 2}`, le tableau des indices de découpage est `{0, 4, 5, 6, 10, 16}` correspondant aux 6 sous-séquences `{1, 4, 5, 8}`, `{7}`, `{5}`, `{2, 3, 4, 9}`, `{3, 3, 3, 5, 5, 7}` et `{2}`. Cette fonction pourra parcourir plusieurs fois de tableau pris en argument, mais ne devra pas allouer d'espace mémoire en dehors de celui nécessaire pour enregistrer le tableau résultat.

R. 4-22 Écrire une fonction permettant de découper un tableau d'entiers en sous-séquences non vides de somme inférieure à 10 maximales pour l'extension à gauche. Par exemple `{1, 2, 8, 2, 3, 4, 1}` est découpé en `{{1, 2}, {8, 2}, {2, 3, 4, 1}}`.

4 Tableaux représentant des matrices

R. 4-23 En utilisant la représentation des matrices par tableau unidimensionnel qui consiste à mettre les lignes bout à bout, écrire une fonction prenant en argument un entier naturel non nul n et retournant un tableau d'entiers unidimensionnel représentant I_n, la matrice identité de dimensions $n \times n$.

R. 4-24 En utilisant la représentation des matrices par tableau unidimensionnel qui consiste à mettre les lignes bout à bout, écrire une fonction prenant en argument deux entiers naturels non nuls n et m, une matrice M de dimensions $n \times m$ à coefficients entiers, et les indices i et j d'un coefficient de M, et qui renvoie le coefficient $M_{i,j}$.

R. 4-25 En utilisant la représentation des matrices par tableau unidimensionnel qui consiste à mettre les lignes bout à bout, écrire une fonction prenant en argument deux entiers naturels non nuls n et m une matrice de dimensions $n \times m$ à coefficients entiers, les indices i et j d'un coefficient de M, et une valeur entière x, et qui donne au coefficient d'indice (i, j) dans cette matrice la valeur x.

R. 4-26 À l'aide de `struct`, définir un type permettant de représenter une matrice d'entiers dans un tableau unidimensionnel qui contient les lignes de cette matrice mises bout à bout.

R. 4-27 Écrire une fonction prenant en argument un entier naturel non nul n et retournant un tableau de tableaux d'entiers représentant la matrice I_n.

R. 4-28 Écrire une fonction prenant en arguments deux entier naturels non nuls n et m un tableau de tableaux d'entiers représentant une matrice M de dimensions $n \times m$, et enfin deux indices i et j valides dans cette matrice et qui renvoie le coefficient $M_{i,j}$.

R. 4-29 Écrire une fonction prenant en arguments deux entier naturels non nuls n et m un tableau de tableaux d'entiers représentant une matrice de dimensions $n \times m$, deux indices i et j valides dans cette matrice et une valeur x, et qui donne au coefficient d'indice i, j dans cette matrice la valeur x.

R. 4-30 À l'aide de `struct`, définir un type permettant de représenter une matrice d'entiers dans un tableau de tableaux d'entiers, dont chaque tableau est une ligne de cette matrice.

5 Tris

R. 4-31 Écrire une fonction permettant de trier un tableau au moyen de l'algorithme du tri à bulles ♣.

R. 4-32 Écrire une fonction permettant de trier un tableau au moyen de l'algorithme du tri par insertion.

R. 4-33 Écrire une fonction permettant de trier un tableau au moyen de l'algorithme du tri par sélection.

R. 4-34 Écrire une fonction partition prenant en arguments un tableau `t` et `ip` un indice de `t` et qui modifie `t` en place de manière à placer d'abord les éléments inférieurs à la valeur pivot `t[ip]`, puis cette valeur pivot, à un indice `q` qu'elle retournera, et enfin ceux strictement supérieurs à cette valeur.

R. 4-35 Écrire une fonction permettant de trier un tableau au moyen de l'algorithme du tri rapide.

6 Autres utilisations classiques des tableaux en algorithmique

R. 4-36 Écrire une fonction prenant en argument un tableau d'entiers, triés par ordre croissant, de taille n, un entier `p` et testant si l'entier `p` apparaît dans le tableau. Cette fonction devra être récursive et ne pas manipuler de boucles `while`/`for`. On assurera une complexité en $\mathcal{O}(\log(n))$.

♣. https://fr.wikipedia.org/wiki/Tri_à_bulles

R. 4-37 Écrire une fonction prenant en argument un tableau d'entiers, triés par ordre croissant, de taille n, un entier `p` et testant si l'entier `p` apparaît dans le tableau. Cette fonction ne devra pas être récursive. On assurera une complexité en $\mathcal{O}(\log(n))$.

R. 4-38 Écrire une fonction prenant en argument un tableau de booléens représentant un entier écrit en base 2 (les bits de poids faibles sont à droite) et modifiant le tableau pour qu'il représente l'entier suivant. Cette fonction renverra un booléen indiquant si l'incrémentation a bien été faite, de sorte qu'on renverra `false` s'il n'est pas possible d'incrémenter l'entier.

R. 4-39 Écrire une fonction permettant de calculer la médiane d'un tableau d'entiers deux à deux distincts sans trier ce tableau. On attend une fonction de complexité pire cas quadratique.

R. 4-40 Écrire une fonction permettant de calculer la médiane d'un tableau d'entiers sans d'abord trier ce tableau. On pourra utiliser la même idée de valeur pivot que dans le tri rapide*Si besoin consulter la deuxième partie du DM sur les k plus proches voisin.* .

R. 4-41 Implémenter une structure de table dynamique, c'est-à-dire une structure de tableau telle que l'ajout et la suppression d'éléments en fin de tableau ont un coût amorti constant.

R. 4-42 On représente une permutation de $[\![0, n-1]\!]$ par un tableau d'entiers `t` tel que l'image de tout entier `i` de $[\![0, n-1]\!]$ est `t[i]`. On représente un cycle (au sens des permutations) par un tableau contenant les éléments de son support, dans l'ordre du cycle, ainsi `{0, 1, 2}` est le cycle qui envoie 0 sur 1, 1 sur 2 et 2 sur 0. On rappelle qu'une permutation se décompose de manière unique en un produit de cycles à supports disjoints. On représente un produit de cycles à supports disjoints comme un tableau des cycles qui le composent. Leurs supports étant disjoints, ces cycles commutent, ainsi l'ordre dans le tableau n'importe pas. Donner une fonction calculant la décomposition d'une permutation en produit de cycles à supports disjoints. On pourra allouer des tableaux temporaires et/ou renvoyer le résultat dans un tableau trop grand muni à la fois de sa taille et de son nombre d'éléments.