

Travaux dirigés n° 4

Tableaux

Exercice 1 (Tableaux - références et passage de paramètres)

1°) Décrivez l'état de la mémoire à chaque instruction de l'algorithme suivant :

```

Algorithme manipulationTableaux
Déclarations
    Variables
        t, u, v : tableaux d'entiers
        i : entier
Début
{1} t ← allouer(3)
{2} t[0] ← 1; t[1] ← 3; t[2] ← 7
{3} u ← t
{4} u[1] ← 9
{5} v ← allouer(3)
{6} Pour i allant de 0 à 2 Faire
{6.1} v[i] ← t[i]
    FinPour
Fin
    
```

À la fin de cet algorithme, déduisez la valeur des expressions suivantes : $u == t$ et $v == t$.

2°) Étant donnée la procédure suivante :

```

Procédure multConst (t : tableau d'entiers, k : entier)
Déclarations
    Variables locales
        i : entier
Début
{1} Pour i allant de 0 à taille(t)-1 Faire
{1.1} t[i] ← t[i] * k
    FinPour
Fin
    
```

Donnez la trace mémoire au fur et à mesure de l'exécution de l'appel `multConst(v,3)` (avec `v` de l'algorithme précédent).

3°) Étant donnée la fonction suivante :

```

Fonction opp (t : tableau d'entiers) : tableau d'entiers
Déclarations
    Variables locales
        taille, i : entiers
        v : tableau d'entiers
Début
{1} taille ← taille(t)
{2} v ← allouer(taille)
{3} Pour i allant de 0 à taille-1 Faire
{3.1} v[i] ← -t[i]
    FinPour
{4} retourner(v)
Fin
    
```

Donnez la trace mémoire de l'exécution de l'instruction : $u \leftarrow opp(v)$ (avec `v` resté dans l'état précédent).

Exercice 2 (Les classiques)

Dans ces questions, on considère des tableaux de réels, préalablement alloués et éventuellement remplis.

Donnez le code des fonctions/procédures réalisant les traitements suivants :

1. Remplissage d'un tableau :
 - toutes les cases sont initialisées à 0 ou à une certaine valeur fixe ;
 - toutes les cases sont initialisées au clavier ;
 - toutes les cases sont initialisées par des valeurs "aléatoires" (les bornes de l'intervalle sont passées en paramètre) en supposant l'existence de la fonction `aleatoire()` : réel retournant un réel aléatoire dans l'intervalle $[0 ; 1]$.
2. Affichage d'un tableau.
3. Recherche du nombre d'occurrences d'une valeur donnée (réelle) dans un tableau.
4. Le maximum :
 - retourne la valeur maximum d'un tableau ;
 - retourne le rang du maximum (on renvoie le premier indice où se trouve cette valeur).
5. Recherche d'un réel donné dans un tableau : on cherche l'indice de sa première occurrence, et on renvoie -1 si la valeur n'est pas présente dans le tableau.
6. On reprend la question précédente, en supposant que le tableau est trié par ordre croissant (par exemple).

Exercice 3 (Autres algorithmes)

Donnez les algorithmes suivants :

1. Fonction `coincidences(u, v : tableaux d'entiers) : entier`
 - Hypothèses : `u` et `v` sont alloués, initialisés et de même taille
 - But : calcule le nombre de cases vérifiant `u[i] == v[i]`
2. Fonction `egal(u, v : tableaux de caractères) : booléen`
 - Hypothèses : `u` et `v` sont alloués, initialisés et de même taille
 - But : teste l'égalité des contenus des deux tableaux
3. Fonction `indiceDiff(u, v : tableaux de caractères) : entier`
 - Hypothèses : `u` et `v` sont alloués, initialisés et de même taille
 - But : retourne le premier indice tel que `u[i] ≠ v[i]` ou -1 s'ils sont égaux
4. Fonction `compare(u, v : tableaux de caractères) : entier`
 - Hypothèses : `u` et `v` sont alloués, initialisés et de même taille
 - But : compare `u` et `v` suivant l'ordre lexicographique ; retourne 0 en cas d'égalité, un résultat strictement négatif si `u < v` ou un résultat strictement positif sinon.
5. Fonction `present(u, v : tableaux de réels) : tableau de booléens`
 - Hypothèses : `u` et `v` sont alloués, initialisés mais pas nécessairement de même taille
 - But : retourne pour chaque case de `u` si la valeur est présente dans `v`
6. Fonction `sousTableau(u, v : tableaux de caractères) : booléen`
 - Hypothèses : `u` et `v` sont alloués, initialisés mais pas nécessairement de même taille
 - But : retourne vrai si les valeurs des cases de `u` sont présentes dans `v` dans le même ordre mais pas forcément de manière contigüe
7. Fonction `sousTableau2(u, v : tableaux de caractères) : booléen`
 - Hypothèses : `u` et `v` sont alloués, initialisés mais pas nécessairement de même taille
 - But : retourne vrai si les valeurs des cases de `u` sont présentes dans `v` dans le même ordre et de manière contigüe

Exercice 4 (Manipulation des binaires)

Donnez les algorithmes suivants :

1. Fonction *binnaire*(*n* : entier) : tableau d'entiers
 - Hypothèses : $n \geq 0$
 - But : calcule et retourne un tableau de 16 cases contenant la représentation binaire de l'entier *n*
2. Procédure *ajouteUn*(*bin* : tableau d'entiers)
 - Hypothèses : *bin* est alloué et initialisé ; il contient des chiffres binaires
 - But : modifie *bin* en lui ajoutant 1
3. Fonction *ajoute*(*bin1*, *bin2* : tableau d'entiers) : tableau d'entiers
 - Hypothèses : *bin1* et *bin2* sont alloués et initialisés ; ils contiennent des chiffres binaires
 - But : retourne la somme des deux binaires
4. Procédure *afficheBin*(*n*, *p* : entiers)
 - Hypothèses : $0 \leq p \leq n$
 - But : affiche les représentations binaires des entiers de *n* à *p*
5. Fonction *complement2*(*bin* : tableau d'entiers) : tableau d'entiers
 - Hypothèses : *bin* est alloué et initialisé ; il contient des chiffres binaires
 - But : retourne le complément à 2 de *bin*
6. Fonction *binToHex*(*bin* : tableau d'entiers) : tableau de caractères
 - Hypothèses : *bin* est alloué et initialisé ; il contient des chiffres binaires
 - But : calcule la représentation hexadécimale à partir de la représentation binaire

Exercice 5 ("Split")

On dispose de trois tableaux de *n* entiers : *t*, *t*₁ et *t*₂. On suppose que *t* est entièrement rempli, et que les cases des deux autres tableaux ont été initialisées à 0.

- 1°) Rappelez les fonctions/procédures qui permettent de réaliser de telles initialisations.
- 2°) Écrivez un algorithme qui répartit dans *t*₁ et *t*₂ les entiers non nuls de *t* : les négatifs dans *t*₁ et les positifs dans *t*₂.
- 3°) Complétez cet algorithme en affichant les contenus des trois tableaux.
- 4°) Même chose, mais sans initialiser les deux tableaux *t*₁ et *t*₂.

Exercice 6 (Triangle de Pascal : 2 versions)

On désire afficher une ligne donnée du triangle de Pascal.

- 1°) Écrivez une procédure permettant d'afficher la ligne *n* du triangle de Pascal. Les lignes sont construites les unes après les autres, en les déduisant les unes des autres.

Vous utiliserez ici deux tableaux *a* et *b* :

- *a* contenant une ligne, calculez la suivante dans *b* ;
- recopiez dans *a* le contenu de *b*...

- 2°) Même question, en utilisant un seul tableau (à une dimension).

Exercice 7 (Pile ou face?)

On désire simuler une suite de lancers d'une pièce et on s'intéresse au nombre de résultats *Pile* ou *Face*. Pour coder ce type de valeurs, on choisit le booléen *faux* pour un résultat *Face* et *vrai* pour un résultat *Pile*.

On suppose disposer de la fonction `aleatoire()` : **réel** qui retourne un réel aléatoire dans l'intervalle $[0; 1[$ et de la fonction `partieEntiere(r : réel)` : **entier** qui retourne la partie entière du réel passé en paramètre.

- 1°) Écrivez une fonction/procédure `lancePiece` qui retourne aléatoirement *vrai* ou *faux*.
- 2°) Écrivez une fonction/procédure `remplitPF` qui remplit aléatoirement le tableau passé en paramètre supposé alloué.
- 3°) Écrivez une fonction/procédure `affiche` qui affiche le contenu du tableau passé en paramètre (supposé alloué et initialisé) : on affichera **P** à la place de *vrai* et **F** à la place de *faux*.
- 4°) Écrivez une fonction/procédure `comptePile` qui calcule et retourne le nombre de cases contenant la valeur *vrai* du tableau passé en paramètre (supposé alloué et initialisé).
- 5°) Écrivez l'algorithme principal qui :
 - fixe un nombre d'essais (choisi aléatoirement entre 10 et 100), et la taille de chaque essai (choisi entre 5 et 20).
 - pour chaque essai :
 - affiche le numéro de l'essai en cours ;
 - remplit un tableau avec le résultat des lancers ;
 - affiche ce tableau ;
 - affiche le nombre de *Pile* et de *Face* obtenus.
- 6°) (*Compléments*) Envisagez (séparément) chacune des modifications suivantes :
 - *Pile* et *Face* sont codés par un caractère.
 - la pièce peut être truquée, on connaît la probabilité d'obtenir le résultat *Pile*.
 - on veut utiliser une fonction de la forme : `construitPF(n : entier) : tableau de booléens` qui construit un tableau de taille n , le remplit et le retourne.
- 7°) (*Et encore...*) Reprendre l'exercice avec un lancer de dé.