

Feuille de TD n° 4

Jérôme Martin

Note : Dans les exercices qui suivent, et sauf indications contraires, il est uniquement demandé d'écrire les procédures ou fonctions. Cependant, les définitions de constantes ou de types nécessaires à ces procédures/fonctions doivent aussi être données.

Exercice 1 Est-ce une voyelle

Écrire la fonction `EstVoyelle` qui, étant donné un caractère, retourne vrai si le caractère est une voyelle et faux dans le cas contraire. Modifier le programme `Javanais` de l'exercice 2 de la feuille n° 3 pour utiliser cette fonction.

Exercice 2 Minimum d'un tableau

Écrire la fonction `Minimum` qui, étant donné un tableau d'entiers, retourne la valeur de l'élément minimum.

Exercice 3 Traduction binaire → décimal

Un nombre binaire codé sur 8 bits peut être vu comme un tableau à une dimension de 8 cases contenant des 0 ou des 1. Par exemple, le nombre 75 (en décimal) dont la représentation binaire est 01001011 peut être vu comme le tableau :

0	1	0	0	1	0	1	1
---	---	---	---	---	---	---	---

1. Écrire la fonction `Bin2Dec` qui traduit un tableau codant un nombre binaire en valeur décimale.

Indication : On pourra utiliser la solution de l'exercice 1 de la feuille 3.

2. Écrire le programme `TraductionBinaire` qui génère un nombre binaire aléatoire, le traduit en valeur décimal à l'aide de la fonction `Bin2Dec` de la question 1 et l'affiche sous la forme :

Le nombre binaire 01001011 est égal à 75 en décimal

Indication : le nombre binaire est généré chiffre par chiffre en utilisant l'appel à la fonction `entier_hasard(0,1)`.

Exercice 4 Recherche d'un élément dans un tableau

1. Écrire la fonction `RechercheElement` qui étant donné un tableau d'entiers et un entier retourne l'indice d'apparition de cet entier dans le tableau s'il existe et `-1` sinon.
2. Modifier la fonction `RechercheElement` pour qu'elle retourne un booléen indiquant si l'entier a été trouvé dans le tableau et un entier correspondant à l'indice d'apparition de cet entier.

Exercice 5 Somme d'une colonne d'un tableau

Écrire la fonction `SommeColonne` qui, étant donné un tableau à deux dimensions d'entiers et un numéro de colonne, calcule la somme des éléments de la colonne.

Exercice 6 Minimum et maximum d'un tableau

Écrire la fonction `MinMax` qui, étant donné un tableau à une dimension d'entiers, calcule et retourne le minimum et le maximum des éléments du tableau.

Indication : Deux éléments ne pouvant être résultats d'une fonction, il faut utiliser des passages par référence (cf. Bertrandias p. 76–77).

Exercice 7 Initialisation aléatoire d'un tableau

Écrire la procédure `InitTableau` qui, étant donné un tableau à deux dimensions de réels passé par référence, remplit le tableau en utilisant la fonction `hasard`.

Exercice 8 Échange de deux éléments dans un tableau

Écrire la fonction `EchangeElementsTableau` qui, étant donné un tableau à une dimension d'entiers et deux indices dans ce tableau, échange les valeurs des éléments de ces deux indices.

Exemple : Si `T` est un tableau défini par :

```
var T : array [1..5] of integer
```

et si `T` est rempli par les valeurs suivantes :

7	1	9	4	0
---	---	---	---	---

Après l'appel de la fonction `EchangeElementsTableau (T,2,4)`, le tableau `T` contient :

7	4	9	1	0
---	---	---	---	---

Exercice 9 Traduction hexadécimal \longleftrightarrow décimal

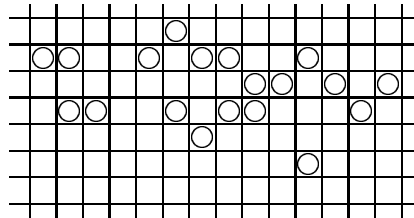
De la même manière que nous l'avons vu dans Exercice 3 pour le codage binaire, un nombre hexadécimale peut être vu comme un tableau à une dimension de 4 cases (dans ce cas, il représente un entier codé sur 32 bits) contenant les chiffres 0 à 9 et les lettres de A à F.

1. Écrire la fonction `Hex2Dec` qui effectue la traduction du tableau contenant le nombre en hexadécimale en une valeur décimale. Nous supposons disposer d'une fonction `ChiffreHex2Dec` effectuant la conversion des chiffres hexadécimaux (0 à 9 et A à F) en valeur décimale. Écrire le prototype de la fonction `ChiffreHex2Dec`.
2. Écrire la fonction `Dec2Hex` qui effectue la traduction inverse. De même, nous supposons disposer d'une fonction de conversion `Dec2ChiffreHexa`. Écrire le prototype de cette fonction de conversion.

Exercice 10 Le jeu de la vie

Le jeu de la vie de J.H. CONWAY est un modèle d'automate cellulaire simple qui permet de déterminer les populations successives d'un tableau binaire bidimensionnel. Il utilise un grille rectangulaire de cellules, chacune d'entre elles pouvant contenir une cellule vivante.

Sur un damier, il y a donc, à un instant donné, certaines cellules vivantes, que nous représentons, dans l'exemple ci-dessous (noté *génération N*), par de petits disques.



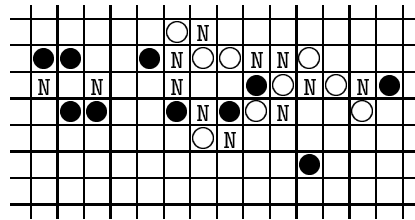
Cette situation va évoluer dans le temps (génération après génération). La naissance, la survie ou la mort d'une cellule dans une case est fonction du nombre de cellules vivantes dans les case voisines (au sens du schéma ci-dessous, où la cellule C possède huit voisines marquées v).

v	v	v
v	C	v
v	v	v

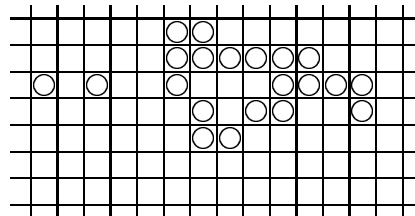
L'état d'une génération est obtenue à partir de la génération précédente en appliquant les règles suivantes :

- dans une case inoccupée, naît une cellule, à la génération suivante, si et seulement si le nombre de cellules vivantes dans les cases voisines est **trois** ;
- dans une case occupée, la cellule survit, à la génération suivante, si et seulement si le nombre de cellules vivantes dans les cases voisines est **deux** ou **trois** (de manière imagée, on dit que la cellule meurt d'isolement — moins de deux voisins — ou d'étouffement — plus de trois.).

La *génération N* va donc évoluer de la façon illustrée par la figure ci-dessous : les cellules représentées par un disque vont disparaître et une nouvelle cellule va naître dans chaque case marquée N.



Ainsi, la *génération N+1* est illustrée par la figure suivante :



1. Écrire la fonction `NombreVoisines` qui détermine le nombre de cellules voisines d'une case (i,j) .
2. Écrire la fonction `Naissance` qui vérifie qu'une case (i,j) est inoccupée et qu'elle remplit la condition pour qu'à la génération suivante une cellule naisse.
3. Écrire la fonction `Survie` qui vérifie qu'une case est occupée et qu'elle vérifie la condition pour que la cellule survive à la génération suivante.
4. Écrire la procédure `Affichage` qui affiche la grille. Une cellule est modélisée par le caractère `o`.

Indications :

1. Pour représenter le quadrillage et l'état de la population à un instant donné, on pourra utiliser un tableau bidimensionnel de booléens.
2. Les lignes et les colonnes extrêmes du tableau seront toujours vides : elles constitueront un bord commode pour évaluer les générations successives mais ne seront jamais considérées comme faisant partie des cases observées.