LIFAP1 – TD 8 : Tableaux 1D et 2D

Objectifs : Apprendre à manipuler les tableaux à deux dimensions et approfondir l'utilisation des tableaux à 1 dimension

Tableaux à 2 dimensions …. ce qui change :	
Déclaration	T : tableau [10] [5] d'entiers (T sera un tableau de 10 lignes et 5 colonnes)
Accès	T[i-1][j-1] (désigne la case à la i^{ème} ligne et j^{ème} colonne)

1. Écrire l'algorithme d'un sous-programme qui calcule et "retourne" un tableau 1D contenant les N premiers termes de la suite U_n définie par :

$$U_0 = 1, U_{n+1} = \frac{U_n}{n+1}$$

```
Procédure suite (T : Tableau [MAX_TAB] de réels, N entier)
  Précondition : aucune
  Données / résultat : T
  Donnée : N entier
  Description : remplit T avec les valeurs de la suite
  Variable locale : i : entier
  Début
    T[0]←1
    Pour i allant de 1 à N-1 par pas de 1 faire
      T[i] ← T[i-1]/ i
    FinPour
  Fin Suite
```

2. Soit T un tableau 2D de taille 5*5 contenant des entiers. Écrire le sous-programme d'initialisation à 0 d'une telle structure de données.

```
Déclaration : T : tableau[5][5] d'entiers
Procédure InitTab(T : tableau[5][5] d'entiers)
Donnée / Résultat : T
Variable locales : i,j : entier
Début
  Pour i allant de 0 à 4 par pas de 1 faire
    Pour j allant de 0 à 4 par pas de 1 faire
      T[i][j]←0
    Fin Pour
  Fin Pour
Fin InitTab
```

3. Écrire un sous-programme `RemplirTab` qui propose à l'utilisateur de remplir un tableau T d'entiers de taille 5*5.

5	1	8	6	0
6	9	7	4	2
1	1	0	9	7
4	5	7	3	0
0	2	5	0	9

```
Procédure RemplirTab(T : tableau[5][5] d'entiers)
Donnée / Résultat : T
Variable locales : i,j : entier
Début
  Pour i allant de 0 à 4 par pas de 1 faire
    Pour j allant de 0 à 4 par pas de 1 faire
      Afficher (« donnez la valeur »)
      Saisir(T[i][j])
    Fin Pour
  Fin Pour
Fin remplirTab
```

4. Écrire deux procédures d'affichage d'un tableau 2D de taille 5*5
    a. `Affichage_2D_ligne` : qui affichera le tableau ligne par ligne

```
Procédure Affiche_2D_ligne(T : tableau[5][5] d'entiers)
Donnée / Résultat : T
Variable locales : i,j : entier
Début
  Pour i allant de 0 à 4 par pas de 1 faire        // ligne
    Pour j allant de 0 à 4 par pas de 1 faire      // colonne
      Afficher (T[i][j])
    Fin Pour                                       // fin colonne
    Afficher (saut de ligne)
  Fin Pour
Fin Affiche_2D_ligne
```

    b. `Affichage_2D_colonne` : qui affichera le tableau colonne par colonne

```
Procedure Affiche_2D_colonne(T : tableau[5][5] d'entiers)
Donnée / Résultat : T
Variable locales : i,j : entier
Debut
  Pour i allant de 0 à 4 par pas de 1 faire        // colonne
    Pour j allant de 0 à 4 par pas de 1 faire      // ligne
      Afficher (T[j][i])
    Fin Pour                                       // fin ligne
    Afficher (saut de ligne)
  Fin Pour
Fin Affiche_2D_colonne
```

5. Écrire une fonction permettant sur un tableau 2D de taille 5*5 de calculer la somme des éléments d'une ligne (le numéro de la ligne étant passé en paramètre).

```
Fonction SommeLigne(T : tableau[5][5] d'entiers, ligne : entier) : entier
Donnée / Résultat: T
Donnée : ligne : numéro de la ligne dont on veut calculer la somme
Résultats : somme des éléments de la ligne "ligne"
Variables locales : i,som_lig : entier
Debut
  som_lig ← 0
  Pour i allant de 0 à 4 par pas de 1 faire
    som_lig ← som_lig + T[ligne][i]
  Fin Pour
  Retourner (som_lig)
Fin SommeLigne
```

6. Soit T un tableau à 2 dimensions de taille M * N contenant des entiers. Ce tableau est rempli avec des nombres sur les L premières lignes et les C premières colonnes. Écrire en langage algorithmique un sous-programme permettant de remplir un tableau 1D avec la somme des colonnes de T. Attention à ne bien parcourir que les colonnes et les lignes remplies.

1	5	6	4	
8	9	0	6	
3	2	7	1	

12	16	13	11	

On commence par définir 2 constantes :
M = 10
N = 10

```
Procedure Remplir_Somme_Colonne(T : tableau[M][N] d'entiers, Tab_Res[N],
Taille_L : entier, Taille_C : entier)
Précondition : Tab_Res initialisé à 0
Donnée / Résultat : T, Tab_Res
Données : Taille_L, Taille_C
Variable locales : i,j : entier
Debut
  Pour j allant de 0 à Taille_C-1 par pas de 1 faire
      Pour i allant de 0 à Taille_L-1 par pas de 1 faire
            Tab_Res[j]=Tab_Res[j] + T[i][j]      Fin Pour
  Fin Pour
Fin Remplir_Somme_Colonne
```

Pour s'entraîner

Écrire une fonction permettant sur un tableau 2D de taille 5*5

a. de calculer la somme des éléments d'une colonne (le numéro de la colonne étant passé en paramètre)

```
Fonction SommeColonne(T : tableau[5][5] d'entiers, colonne : entier) : entier
Donnée / Résultat: T
Donnée : colonne : numéro de la colonne dont on veut calculer la somme
Résultats : somme des éléments de la colonne "colonne"
Variables locales : i,som_col : entier
Debut
  som_col ← 0
  Pour i allant de 0 à 4 par pas de 1 faire
    som_col ← som_col + T[i][colonne]
  Fin Pour
  Retourner (som_col)
Fin SommeColonne
```

b. de calculer les sommes des éléments de chaque diagonale (dans la mesure où le tableau est bien carré)

```
Fonction SommeDiagonale(T : tableau[5][5] d'entiers, som_diag2 : entier) : entier
Précondition : le tableau T est carré
Donnée / Résultat: T, som_diag2
Résultats : somme des éléments de la première diagonale
Variable slocales : i,som_diag1 : entier
Debut
som_diag ← 0
som_diag2 ← 0
  Pour i allant de 0 à 4 par pas de 1 faire
    som_diag1 ← som_diag1 + T[i][i]
    som_diag2 ← som_diag2 + T[i][4-i]

  Fin Pour
  Retourner (som_diag1)
Fin
```