# Les boucles et les tableaux

TD2

Avant chaque écriture de pseudo-code il faut suivre ces étapes successivement :

a) Faire des réflexions sur le problème à résoudre.
b) Etudier la validation des données d'entrée.
c) Etudier le cas général et les cas particuliers à traiter.
d) Ecrire une 1ére version l'algorithme en Pseudo-code.
e) Simuler l'exécution de l'algorithme avec des valeurs assez variées.
f) S'il ne donne pas les résultats attendus, revenir à (a)
g) Donner la version finale

Sur papier ou sur ordinateur, respectez les règles est les conventions pour la clarté de votre pseudo-code

## Exercice 1 :

Ecrire un algorithme qui demande deux entiers à l'utilisateur et qui calcule la somme des entiers compris entre ces deux valeurs. Par exemple, si l'on entre 5 et 12, le programme doit afficher **68** :
5 + 6 + 7 + 8 + 9 + 10 + 11 + 12 = **68**

## Exercice 2 :

Ecrire un algorithme qui affiche les 9 tables de multiplication pour les entiers de 1 à 9. Chaque table comporte 9 éléments, sous la forme suivante :

```
1  2  3  4  5  6  7  8  9
2  4  6  8 10 12 14 16 18
.
.
.
9 18 27 36 . . .
```

## Exercice 3 :

Ecrire un algorithme qui saisit un entier et qui l'affiche à l'envers.
Par exemple, l'utilisateur saisit 123456 et le programme affiche 654321.

## Exercice 4 :

Ecrire un algorithme qui demande à l'utilisateur de fournir deux nombres entiers n et p, tel que $\boldsymbol{n \geq p}$. Puis, il calcul le nombre de combinaisons de p par n : $\boldsymbol{C_n^p}$

## Exercice 5 :

Ecrire un algorithme qui affiche le nombre de chiffre dans nombre entier N (positive ou négative) introduit par l'utilisateur.

Exemple d'exécution :

```
> Introduire un nombre: 123

> Le nombre 123 a 3 chiffres.

> Introduire un nombre: 580499

> Le nombre 580499 a 6 chiffres.
```

# Exercice 6 :

Pour sa naissance, la grand-mère de Karim place une somme de 10000dh sur son compte épargne rémunéré au taux de 2.25% (chaque année le compte est augmenté de 2.25%).

Développer un algorithme permettant d'afficher le montant de son compte lorsque Karim va atteindre un certain âge.

# Exercice 7 :

Ecrire un algorithme qui demande successivement 20 nombres à l'utilisateur, et qui lui dise ensuite quel était le plus grand parmi ces 20 nombres :

```
Entrez le nombre numéro 1 : 12
Entrez le nombre numéro 2 : 14
…
Entrez le nombre numéro 20 : 6
Le plus grand de ces nombres est  : 14
```

Modifiez ensuite l'algorithme pour que le programme affiche de surcroît en quelle position avait été saisie ce nombre :

```
C'était le nombre numéro 2
```

# Exercice 8 :

Soit les suites numériques récurrentes définies comme suit :

1. $\begin{cases} U_0 = 1 \\ U_n = 6U_{n-1} + 4 \quad \forall n \in \mathbb{N}, n > 0 \end{cases}$
2. $\begin{cases} U_0 = 0, U_1 = 1 \\ U_n = 6U_{n-1} - 9U_{n-2} \quad \forall n \in \mathbb{N}, n > 1 \end{cases}$
3. $\begin{cases} U_0 = 4, \\ \forall n \in \mathbb{N}, n > 0 \begin{cases} U_n = \frac{U_{n-1}}{2} \; si \; n \; est \; pair \\ U_n = \frac{U_{n-1}+1}{2} \; si \; n \; est \; impair \end{cases} \end{cases}$

Ecrire les algorithmes qui calculent et affichent la valeur de chacune de ces suites pour une certaine valeur de « ***n*** » introduite par l'utilisateur.

# Exercice 9 :

Soit les séries numériques définies comme suit :

1. $\sum_{i=1}^{n} i^2$
2. $\sum_{i=1}^{n} \sum_{j=0}^{m} \frac{1}{i+j}$
3. $\sum_{i=1}^{n} (\sum_{j=0}^{i} (\sum_{k=j}^{i} (i + j - k) + 4) \times 2)$
4. $\prod_{i=1}^{n} \sum_{j=1}^{m} \frac{j+i}{i*j}$

Ecrire les algorithmes qui calculent et affichent la valeur de chacune de ces séries pour une certaine valeur de « ***n*** » et « ***m*** » introduite par l'utilisateur.

# Exercice 10 :

1. Ecrire un algorithme qui calcul tous les diviseurs d'un entier ***n*** introduit par l'utilisateur.
2. Ecrire un algorithme qui détermine si un nombre ***n*** est un carrée parfait.
3. Ecrire un algorithme qui détermine si un nombre ***n*** est un nombre premier.

# Exercice 11 :

Ecrire un algorithme qui permet de calculer et d'afficher la représentation binaire d'un nombre naturel introduit par l'utilisateur :

- Sans l'utilisation des tableaux
- Avec l'utilisation des tableaux

## Exercice 12 :

Ecrie un algorithme qui permet d'inverser l'ordre des éléments d'un tableau d'entiers en plaçant le dernier élément en premier et ainsi de suite, voir la figure suivante :

Tableau initial :

| 55 | 39 | 47 | 5 | 17 | 29 | 37 | 4 |
|---|---|---|---|---|---|---|---|

Tableau après l'inversion de l'ordre :

| 4 | 37 | 29 | 17 | 5 | 47 | 39 | 55 |
|---|---|---|---|---|---|---|---|

Vous devez tenir compte que le nombre et les éléments du tableau doivent être introduits par l'utilisateur. L'algorithme doit aussi afficher le tableau inversé.

## Exercice 13 :

Soit un tableau T à deux dimensions (22, 18) que vous devrez remplir de valeurs numériques réels.
Écrire un algorithme qui :

- Recherche et affiche la plus grande valeur de chaque ligne
- Recherche et affiche la plus petite valeur de chaque colonne.
- Recherche la plus grande valeur et la plus petite entre toutes les valeurs de ce tableau.

## Exercice 14 :

Etant donnée un tableau T de 40 entiers, écrire un algorithme permettant de :

- Saisir le tableau T.
- Calculer la somme et la moyenne de T.
- Chercher une valeur donnée dans le tableau et afficher sa position dans le tableau.
- Calculer le nombre d'occurrence d'une valeur donnée.
- Afficher tous les éléments pairs de T.

## Exercice 15 :

Soient deux vecteurs (tableaux) U et V ayant le même nombre d'éléments réels saisies au clavier.
Ecrire un algorithme permettant de :

- Calculer le vecteur somme de U et V.
- Calculer la norme de chacun des vecteurs U et V.
- Calculer le produit cartésien de U et V.

## Exercice 16 :

Ecrire un algorithme permettant de résoudre le problème suivant :
Etant donné un tableau *tab* contenant 100 entiers remplis par l'utilisateur, l'algorithme affichera "vrai" si le tableau est trié du plus petit au plus grand et "faux" sinon.

## Exercice 17 :

Ecrire des algorithmes permettent de calculer à partir de matrice A et B introduites par l'utilisateur :

1. La somme A+B
2. Le produit de convolution $A \otimes B$
3. Le produit matriciel $A \times B$

**NB : pour chaque calcul matriciel, une réflexion sur les tailles des matrices est exigée.**

# Exercice 18 :

Ecrire des algorithmes permettent de remplir automatiquement les matrices des types suivants :

1. Matrice carrée nulle de taille ***n*** (***n*** introduit par l'utilisateur).
2. Matrice identité de taille ***n.***
3. Matrice tri-diagonale de taille ***n*** du type (exemple pour n =6) $M_6 = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 3 & 1 & 0 & 0 \\ 0 & 0 & 1 & 4 & 0 & 0 \\ 0 & 0 & 0 & 0 & 5 & 1 \\ 0 & 0 & 0 & 0 & 1 & 6 \end{bmatrix}$.

# Problème : Matrice nilpotente

Une matrice nilpotente est une matrice dont il existe une puissance égale à la matrice nulle. Elle correspond à la notion d'endomorphisme nilpotent sur un espace vectoriel de dimension finie.

**Définition :**

On dit qu'une matrice carrée $\boldsymbol{A}$ non nulle est **nilpotente** s'il existe un entier naturel $p$ tel que la matrice $A^p$ soit nulle.
**L'indice de nilpotence** est alors le plus petit **p** tel que $A^p$ soit nulle.

**Exemple :**

Considérons la matrice de taille 3x3 suivante :

- $M = \begin{bmatrix} 3 & 9 & -9 \\ 2 & 0 & 0 \\ 3 & 3 & -3 \end{bmatrix}$

En calculant la représentation matricielle de $M^2$ puis celle de $M^3$, on trouve

- $M^2 = \begin{bmatrix} 0 & 0 & 0 \\ 6 & 18 & -18 \\ 6 & 18 & -18 \end{bmatrix}$ et $M^3 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$

Donc l'indice de nilpotence est égal à 3.

On se propose alors d'écrire un algorithme qui permet de calculer cet indice de nilpotence d'une matrice introduite par l'utilisateur. Les étapes clés de cet algorithme sont les suivantes :

- Calcul des puissances successives d'une matrice donnée.

  Ecrire un algorithme qui permet de calculer d'une matrice M à la puissance n
- Vérification si une matrice donnée est nulle

  Ecrire un algorithme qui permet de vérifier si une matrice est nulle

L'algorithme doit répéter ces étapes jusqu'à ce que la puissance de la matrice introduite devienne nulle, si c'est le cas nous venons de vérifier que la matrice est nilpotente et ensuite nous affichons l'indice de nilpotence (égale à cet indice quand la puissance de la matrice devienne nulle).
Dans le cas où la matrice introduite n'est pas nilpotente, les puissances successives ne seront jamais nulles. Donc quand est ce que nous pouvons dire qu'une matrice n'est pas nilpotente ?
Il existe une propriété qui dit que :

**L'indice d'une matrice nilpotente, si il existe, il est égal à la dimension de sa plus grande matrice de Jordan**

Nous n'allons pas faire une démonstration mathématique de cette propriété, et en même temps cette règle semble plus complexe à l'exécuter algorithmiquement.
Nous allons conjoncturer une règle plus simple :

**L'indice d'une matrice nilpotente est inférieur ou égal à la dimension de la matrice**

Donc dès que le nombre de multiplications successives de la matrice dépasse sa dimension, on arrête le calcul est nous affichons un message disant : la matrice introduite n'est pas nilpotente.

- Sur la base de cette étude, écrire un algorithme qui permet de savoir si une matrice carrée introduite par l'utilisateur est nilpotente ou pas, et puis afficher **l'indice de nilpotence** si c'est ce cas.