**Université Moulay Ismail** **Année universitaire : 2024/2025**
**Faculté des sciences de Meknès** **Filière : MIP**
**Département d'Informatique** **Module : INFO2**

# TD N°2 : Fonctions et procédures récursives

## Exercice 1 :

Ecrire une **fonction récursive** qui calcule la somme de N premiers nombres entiers naturels:

S=1+2+3+……+(N-1)+N

## Solution :

Pour écrire la forme **récursive** de la **fonction** somme, il faut chercher tout d'abord la récurrence mathématique.

S(0) = 0
S(N)=N+S (N-1)

```
Algorithme Calcul_Somme
Variables
        S, X : entier
Fonction somme (N : entier) : entier
        Debut
                Si (N=0) alors
                        Retourner (N)
                Sinon
                        Retourner (N+somme(N-1))
                Finsi
        Fin
Début
        Ecrire ("Entrez une valeur entière")
        Lire (X)
        S←somme(X)
        Ecrire("La somme de X=", X, " est : ", S)
Fin
```

## Programme en Python :

```
def somme (N: int):
   if N==0 :
      return 0
   else :
      return (N+somme(N-1))

N=int(input("Entrer une valeur entière"))
while(N<0):
        N=int(input("RE-entrer une valeur entière positive"))
   print("La somme de N premiers nombres entiers naturels : ", somme(N))
```

## Exercice 2 :

Écrire une fonction récursive qui calcule le produit de deux entiers positifs.

```
Fonction produit(a : entier, b : entier) :  entier
Début
  Si ( a = 0 ) alors
    Retourner (0)
  Sinon
    Retourner ( produit(a − 1, b) + b)
  FinSi
Fin
```

## Exercice 3 :

Ecrire une procédure récursive qui permet d'afficher la valeur binaire d'un entier n.

```
Procédure binaire (n : entier )
 Si (n>=1) alors
   binaire (n/2)
   Ecrire (n mod 2)
 FinSi
Fin
```

## Exercice 4 :

Ecrire une procédure récursive qui affiche tout le contenu d'un tableau T de n entiers.

```
Procédure Afficher (i: entier, Tableau T: entier, n:entier)
   Si (i <= n)  alors
      Ecrire(T[i])
      Afficher(i+1, T, n)
   FinSi
Fin
```

---

```
Procédure Afficher (Tableau T: entier, N:entier)
   Si (N >0)  alors
      Afficher(T, N-1)
      Ecrire(T[N])
   FinSi
Fin
```

## Exercice 5 :

Ecrire une **fonction récursive** qui calcule le $N^{ieme}$ terme da la suite numérique définie par :

$U_0 = 2$

$U_1 = 2$

$U_2 = 2$

$U_n = 6*U_{n-1} + 4*U_{n-2} - 5*U_{n-3}$ pour tout $n > 2$

**Solution :**

```
Fonction suite (N : entier) : entier
        Début
                Si N<= 3 alors
                        Retourner (2)
                Sinon
                        Retourner (6*suite(N-1) + 4*suite(N-2) - 5*suite(N-3))
                Finsi
        Fin
```

## Programme en Python :

```
def suite(N: int):
   if(N<=2) :
      return 2
   else :
      return (6*suite(N-1)+4*suite(N-2)-5*suite(N-3))

N=int(input("Entrer une valeur entière"))
while(N<0):
   N=int(input("RE-entrer une valeur entière positive"))
print("Le Nieme terme da la suite numérique est: ", suite(N))
```

## Exercice 6 :

On appelle palindrome une chaîne de caractère qui donne la même chaîne selon que l'on la lire de gauche à droite ou inversement. Autrement dit, le premier caractère est égal au dernier caractère, le deuxième caractère est égal à l'avant dernier caractère, etc.
Une définition récursive d'un palindrome est :

- La chaîne vide est un palindrome.
- La chaîne constituée d'un seul caractère est un palindrome.
- aXb est un palindrome si a = b et si X est un palindrome.

Écrire une fonction récursive qui teste si une chaîne de caractères est palindrome ou non. Cette fonction renvoie Vrai si elle palindrome et Faux si elle ne l'est pas.

**Solution :**

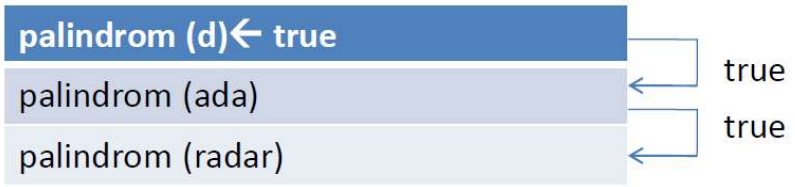
## Exemple : radar

1. Décrire la condition d'arrêt : quand peut-on trouver facilement la solution ?
   **La chaîne constituée d'un seul caractère est un palindrome ou de 0 caractère est palindrome**
2. Réduire le problème à un problème d'ordre inférieur pour que la condition d'arrêt soit atteinte un moment donné
   **a X b est un palindrome si a = b et si X est un palindrome.**

**Cas de base :** La chaîne constituée d'un seul caractère est un palindrome

**Cas général :** aXb est un palindrome si a = b et si X est un palindrome.

Si (a=b) alors
    palindrome (**"radar"**) ← palindrom("ada")
Sinon
    Retourner (false)
Si a=b alors
    palindrom("ada" ) ← palindrom( "d" )
Sinon
    Retourner (false)
palindrom(d) ← true




**Récursivité terminale**

```
Fonction palindrome (tableau ch: chaine de caractère, entier: i, entier j) : booléen
    Début
        Si ( i ≥ j) alors
            Retourner Vrai
        Sinon
            Si (ch[i] = ch[j]) alors
                Retourner (palindrome(ch, i + 1, j –1))
            Sinon
                Retourner Faux
            FinSi
        FinSi
     Fin
```

**Programme en python**

```
def palindrome (ch, i, j) :
   if(i>=j) :
      return True
   elif (ch[i]==ch[j]):
      return (palindrome(ch, i+1, j-1))
   else:
      return False
```

```
mot=input("Entrer une valeur entière")
print("Le mot", mot," est Palindrome : ", palindrome(mot,0, len(mot)-1))
```

## Exercice 7 :

1. Écrire de manière récursive une fonction qui donne le plus grand commun diviseur (pgcd) de deux entiers.
2. Trouver les relations de récurrence en utilisant une méthode de détermination du pgcd basée sur la soustraction

## Solution :

- **En informatique la récursivité consiste à remplacer une boucle par un appel à la fonction elle-même.**

- On va créer une fonction qui pour fournir son résultat, elle va s'appeler elle-même un certain nombre de fois. Cet appel de la fonction par elle-même est la récursivité. Toutefois, il est impératif que ces auto-appels de la fonction PGCD s'arrêtent.

1. Décrire la condition d'arrêt : quand peut-on trouver facilement la solution ?
   **Cas de base (a=b) :**
   **Si a=b le PGCD c'est a ou b donc on arrête donc retourner (a) ou retourner (b)**
2. Réduire le problème à un problème d'ordre inférieur pour que la condition d'arrêt soit atteinte un moment donné
   **Cas général (a>b) :**

```
Si (a > b) Alors
        Resultat← PGCD(a -b, b)
Sinon
        Resultat← PGCD(a, b -a)
FinSi
        Retourner (Resultat)
```

A=90 B=36 → PGCD (90,36)→PGCD (90-36=54, 36)

A=54 B=36 → PGCD (54-36=18, 36)

A= 18 B=36 →PGCD (18, 36-18=18)

A=18 B=18 → A=18

PGCD (18, 18)→ 18

PGCD (18, 36)

PGCD (54, 36)

PGCD (90, 36)

# Récursivité terminale

```
Fonction PGCD (a, b : entier) : entier
        Variable
                résultat : entier
        Debut
                Si (a = b) Alors
                        résultat = a
                Sinon Si (a > b) Alors
                                résultat = PGCD(a -b, b)
                        Sinon
                                résultat = PGCD(a, b -a)
                        FinSi
                FinSi
                Retourner (résultat)
        Fin
```

**Programme en Python :**

```
def PGCD(a,b):
    if(a==b) :
        resultat=a
    elif (a>b):
        resultat= PGCD(a-b,b)
    else:
        resultat= PGCD(a, b-a)
    return resultat

a=int(input("Entrer une valeur entière a="))
b=int(input("Entrer une valeur entière b="))
print("Le PGCD est : ", PGCD(a,b))
```