LIFAPI – TD 2 : Algorithmes un peu moins simples

Objectifs : Approfondir les notions vues dans le TD précédent (boucles, conditions, structures de données, entrées / sorties, …)

```
Conditionnelle à choix multiple (sélective)

SELON (sélecteur) FAIRE
              Cas <liste de valeurs-1> : <suite d'action (s)-1>
              [Cas <liste de valeur-2> : <suite d'action (s)-2>
               ……. ]
              [Autrement : <suite d'action (s)-n> ]
FINSELON

Le sélecteur est une variable de type entier ou caractère
```

1. Écrire l'algorithme d'un programme permettant de simuler le fonctionnement d'une calculatrice simple (+, -, *, /). Dans cet exercice, l'utilisateur saisira les deux opérandes, l'opérateur et le programme lui affichera le résultat correspondant. Dans le cas d'une division, on vérifiera bien que le dénominateur est non nul !
   a. avec des `si` imbriqués
   b. avec un sélecteur `selon`

Avec des si imbriqués

```
Début

    Nb1, Nb2 : reels
    Op : caractère
    Afficher('Donnez le premier nombre :')
    Saisir(Nb1)
    Afficher('Donnez le deuxième nombre :')
    Saisir(Nb2)
    Afficher('Donnez le symbole de l'opération :')
    Saisir(Op)
    Si Op =  '+' alors Afficher (Nb1 + Nb2 )
          Sinon si op= '-'  alors Afficher (Nb1 - Nb2 )
                Sinon si op= '*' alors Afficher (Nb1 * Nb2)
                      Sinon si op= '/' alors
                                  Si (Nb2 = 0)  Alors Afficher ('Opération impossible')
                                  Sinon Afficher( Nb1 / Nb2)
                                  Fin si
                                sinon Afficher ('Opération inexistante')
                           Fin si
                     Fin si
                Fin si
    Fin si
Fin
```

Avec un sélecteur

```
Début

    Nb1, Nb2 : reels
    Op : caractère
    Afficher('Donnez le premier nombre :')
    Saisir(Nb1)
    Afficher('Donnez le deuxième nombre :')
    Saisir(Nb2)
    Afficher('Donnez le symbole de l'opération :')
    Saisir(Op)
```

```
        Selon Op
          '+' : Afficher (Nb1 + Nb2 )
          '-' : Afficher (Nb1 - Nb2 )
          '*' : Afficher (Nb1 * Nb2)
          '/' :  Si (Nb2 = 0)   Alors Afficher ('Opération impossible')
                                Sinon Afficher( Nb1 / Nb2)
                 Fin si
          Autrement : Afficher ('Opération inexistante')
        Fin Selon
    Fin
```

2. Écrire un algorithme permettant de calculer la factorielle d'un entier n donné par l'utilisateur. On écrira une version avec une boucle conditionnelle et une avec une boucle inconditionnelle.
   Exemple : valeur saisie : 6 ➔ résultat : `720 (= 1 * 2 * 3 * 4 * 5 * 6)`

Avec une boucle inconditionnelle

```
Début
  Variables : n, factorielle, i : entier
  Afficher('donnez la valeur de n')
  Saisir(n)                    // on ne vérifiera pas que n >0
  factorielle ←1
  Pour i allant de 1 à n par pas de 1 faire // on peut commencer à 2
    factorielle ← factorielle * i
  Fin Pour
  Afficher (factorielle)
Fin
```

Avec une boucle conditionnelle

```
Début
  Variables : n, factorielle, i : entier
  Afficher('donnez la valeur de n')
  Saisir(n)                    // on ne vérifiera pas que n >0
  factorielle ←1
  i←1
  Tant que i<=n faire
    factorielle ← factorielle * i
    i←i+1
  Fin tant que
  Afficher (factorielle)
Fin
```

3. Écrire un algorithme permettant de calculer la somme des n premiers nombres impairs.
   Exemple : valeur saisie : 6 ➔ résultat : `36 (= 1 + 3 + 5 + 7 + 9 + 11)`

```
Début
  Variables : n, somme, i : entier
  Afficher('donnez la valeur de n')
  Saisir(n)
  somme←0
  Pour i allant de 1 à 2*n par pas de 2 faire
    somme← somme + i
  Fin Pour
  Afficher (somme)
Fin
```

Exemple pour n=5
Somme = 1+3+5+7+9 = 25 = 5²
Donc la somme des n premiers nombres impairs est égale au carré de n.

4. Écrire un algorithme permettant d'afficher toutes les combinaisons possibles de valeurs sur 2 dés. Résultat : `1-1, 1-2, 1-3, 1-4, 1-5, 1-6, 2-1, 2-2, … , 6-5, 6-6`.

```
Début
   // Déclaration des variables
   de1, de2 : entier

   // Boucles imbriquées pour générer les combinaisons
   pour de1 allant de 1 à 6 par pas de 1 faire
      pour de2 allant de 1 à 6 par pas de 1 faire
         afficher (dé1, "-", dé2)
      fin pour
   fin pour
Fin
```

5. Modifier le programme précédent pour ne pas afficher les doublons (`1-2` et `2-1` par exemple).

```
Début
   // Déclaration des variables
   de1, de2 : entier

   // Boucles imbriquées pour générer les combinaisons
   pour de1 allant de 1 à 6 par pas de 1 faire
      pour de2 allant de de1 à 6 par pas de 1 faire
         afficher (dé1, "-", dé2)
      fin pour
   fin pour
Fin
```

6. Écrire un algorithme qui calcule la moyenne de `n` valeurs saisies par l'utilisateur, `n` étant choisi préalablement par l'utilisateur. On recommencera la saisie de `n` tant qu'il n'est pas strictement positif.

```
Début
   n, i : entier    // Déclaration des variables
   valeur, somme, moyenne : réel
   somme ← 0    // Initialisation
   faire    // Saisie de n avec vérification
      afficher "Combien de valeurs voulez-vous saisir ?"
      lire (n)
   tant que n <= 0
   pour i allant de 1 à n par pas de 1 faire // Saisie des n valeurs et calcul de la somme
      afficher ("Entrez la valeur ", i)
      lire (valeur)
      somme ← somme + valeur
   fin pour
   moyenne ← somme / n       // Calcul de la moyenne
   afficher ("La moyenne est : ", moyenne)    // Affichage du résultat
Fin
```

Note : on peut compacter les 2 dernières instructions en afficher ("La moyenne", somme / n)

Exercices pour aller plus loin …

1. Écrire un algorithme permettant de lire 20 nombres entiers au clavier. Si le nombre x saisi est pair, on affiche la valeur (x / 2) sinon on affiche (3*x + 1). Attention, on ne mémorisera pas les 20 valeurs saisies.

```
Puisque l'on connaît le nombre de passages dans la boucle, on utilise la boucle pour :
Début
  Variables : nbre, i : entier
  Pour i allant de 1 à 20 par pas de 1 faire
        Afficher ('Entrez un nombre')
        Saisir(nbre)
        Si (nbre modulo 2) = 0      Alors    nbre ← nbre / 2
                                    Sinon    nbre ← 3*nbre + 1
        FinSi
        Afficher(nbre)
  FinPour
Fin
```

2. Afficher tous les nombres pairs compris entre 0 et 20 inclus
   a. en utilisant une boucle `pour`
   b. en utilisant une boucle `tant que`

a- en utilisant une boucle pour

```
Début
  i : Entier
  Pour i allant de 0 à 20 par pas de 2 faire
    Afficher(i, ' ')
  Fin Pour
Fin
```

```
Début
  i : Entier
  Pour i allant de 0 à 10 par pas de 1 faire
    Afficher(i*2, ' ')
  Fin Pour
Fin
```

b- en utilisant une boucle tant que

```
Début
i : entier
i ← 0
  Tant que  (i ≤ 20)
    Afficher(i, ' ')
    i ← i + 2
  Fin tant que
Fin
```

```
Début
i : entier
i ← 0
  Tant que  (i ≤ 10)
    Afficher(i *2, ' ')
    i ← i + 1
  Fin tant que
Fin
```