LIFAPI – TD 4 : Les sous-programmes

Objectifs : Assimiler la différence entre une fonction et une procédure
Savoir déclarer et utiliser un sous-programme

Recommandations :

Pour chaque **algorithme** demandé, vous préciserez (en justifiant) s'il s'agit d'une **procédure** ou d'une **fonction**.

Vous écrirez **l'entête du sous-programme** (sans oublier les préconditions, les données et résultats, les déclarations des variables locales…) ainsi qu'un **exemple d'appel au sous-programme**.

```
Fonction nom(liste paramètres):type retourné
  Préconditions :
  Données :
  Résultat :
  Description :
  Variables locales :
Début
  instruction(s)
  retourner valeur (ou renvoyer)
Fin nom

Appel : variable ← nom (paramètres)
Exemple : f ← factorielle (6)
```

```
Procédure nom (liste des paramètres)
  Préconditions :
  Données :
  Description :
  Variables locales :

Début
  instruction(s)
Fin nom


Appel : nom (paramètres)
Exemple : mention  (12)
```

1. Rappeler en quelques mots la différence entre une fonction et une procédure. Donner un exemple caractéristique pour chaque.

   Fonction : renvoie un résultat mais ne modifie pas l'environnement
   exemple : factorielle
   Procédure : ne renvoie rien mais modifie l'environnement.
   exemple : affichage_mention
   Faire quelques rappels de cours sur :
   - en-tête fonction / procédure
   - paramètres formels / effectifs,
   - appel d'une fonction (affichage du résultat, affectation, comparaison, …),
   - appel d'une procédure (ce qu'on ne peut pas faire).

2. Écrire l'algorithme d'un sous-programme qui retourne la moyenne de deux réels a et b donnés en paramètre. Écrire le programme principal qui utilise le sous-programme précédent et affiche le résultat produit.

   Fonction moyenne (a : réel, b : réel) : réel
   Précondition : aucune
   Données : a et b
   Résultat : moyenne de a et b
   Description : fonction qui calcule la moyenne de deux réels
   Variable locale : c : réel
   Début
     C ← (a+b) / 2
     Retourner c
   Fin moyenne

   Début
     Variables locales : v1, v2, res : réels
     Afficher ('première valeur :')
     Saisir (v1)
     Afficher ('deuxième valeur :')
     Saisir (v2)
     res = moyenne (v1,v2)   // ou Afficher (moyenne(v1,v2))
     Afficher (res)
   Fin
   Commencez à parler de paramètres **formels** (a et b) et **effectifs** (v1 et v2);
   insistez sur le fait qu'ils portent des **noms différents**.

3. Écrire l'algorithme d'un sous-programme qui affiche les dix nombres suivants la valeur n donnée en paramètre. Par exemple, si l'utilisateur entre le nombre 17, le programme affichera les nombres de 18 à 27. Écrire le programme principal qui utilise le sous-programme précédent.

```
Procédure  suite (n : entier)
Précondition : aucune
Données : n
Description : Affiche les 10 valeurs suivant n
Variable locale : i : entier
Début
  Pour i allant de n+1 à n+10 par pas de 1 faire
    Afficher(i, ' ')
  Fin pour
Fin suite

Appel :
Début
  Variables locales : val : entier
  Afficher('donnez votre valeur :')
  Saisir(val)
  suite(val)
Fin
```

4. Écrire l'algorithme d'un sous-programme qui calcule et retourne la somme des n premiers entiers. Rappel : $1 + 2 + 3 + \ldots + n = n*(n+1) / 2$
   Écrire le programme principal qui utilise le sous-programme précédent.

```
Fonction sommeN (n : entier) : entier
Précondition : n >= 0
Donnée : n
Résultat : somme des n premiers entiers naturels
Variables locales : som, i : entier
Début
  som ← 0
  i ← n
  Tant Que i > 0 Faire
    som ← som + i
    i ← i - 1
  FinTantQue
  Retourner som
Fin

Deuxième version
Fonction sommeN (n : entier) : entier
Précondition : n >= 0
Donnée : n
Résultat : somme des n premiers entiers naturels
Début
  Retourner (n * (n + 1)) / 2
Fin
```

Évitez de leur parler de la version récursive, ça sera fait dans en LIFAPR.

```
Appel : (commun aux deux versions)
Début
  Variables locales : val, res : réels
  Afficher ('Valeur jusqu'à laquelle on veut calculer la somme :')
  Saisir (val)
  res ← sommeN (val)
  Afficher (res)
Fin
```

5. Un nombre **parfait** est un nombre naturel n non nul qui est égal à la somme de ses diviseurs stricts (n exclus). Exemple : 6 = 1 + 2 + 3
    a. Écrire en langage algorithmique une fonction booléenne qui retourne vrai si un entier n passé en paramètre est un nombre parfait, faux sinon.
    b. Écrire en langage algorithmique le programme principal permettant d'afficher la liste des nombres parfaits compris entre 1 et 10000. On utilisera le résultat renvoyé par la fonction précédente.

```
Fonction parfait (n : entier) : booléen
Précondition : n > 0
Donnée : n
Résultat : booléen
Description : retourne vrai si n est parfait, faux sinon
Variable locale : res : booléen, i, som : entiers
Début
  som ← 0
  Pour i allant de 1 à n-1 par pas de 1 faire
       Si (n modulo i ) = 0  Alors som ← som + i
       Fin si
  Fin pour
  Retourner ( n = som)
Fin

Remarque : dès que som > n, on peut s'arrêter avec un tantque

Appel :
Début
  i : entier
  pour i allant de 1 à 10000 par pas de 1 faire
    si parfait(i) alors alors afficher(i, " est parfait.")
    fin si
  fin pour
fin
```