

Séance 4 : Vendredi 08/02/2019

Exercices TD3 : fonctions et procédures

1. Rappeler en quelques mots la différence entre une fonction et une procédure. Donnez 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

Appel :

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.

```

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 demande à l'utilisateur et retourne au programme principal une valeur entière comprise entre 0 et 20. La saisie sera recommencée tant que la valeur choisie n'appartient pas à l'intervalle [0 ; 20].

```

Fonction saisie_bornee () : entier
Précondition : aucune
Donnée : aucune
Résultat : valeur entière comprise entre 0 et 20
Variables locales : valeur : entier
Début
    Faire
        Afficher ("donnez une valeur entière comprise entre 0 et 20")
        Saisir (valeur)
    Tant que ((valeur < 0) ou (valeur > 20))
        Retourner (valeur)
Fin

```

Deuxième version :

```

Fonction saisie_bornee () : entier
Précondition : aucune
Donnée : aucune
Résultat : valeur entière comprise entre 0 et 20
Variables locales : valeur : entier
Début
    Afficher ("donnez une valeur entière comprise entre 0 et 20")
    Saisir (valeur)
    Tant que ((valeur < 0) ou (valeur > 20))
        Afficher ("la valeur doit être comprise entre 0 et 20")
        Saisir (valeur)
    Fin tant que
    Retourner (valeur)
Fin

```

```

Appel (commun au deux versions) :
    Début
        variables locales : note : entier
        note ← saisie_bornee()
    fin

```

5. Écrire l'algorithme d'un sous-programme qui calcule la somme des n premiers entiers. Rappel : $1 + 2 + 3 + \dots + n = n(n+1)/2$

```
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
```

```
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
```

6. 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.

```
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

- 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.

```
Début
  Variables locales : i : entier
  Pour i allant de 1 à 10000 par pas de 1 faire
    Si parfait(i)      Alors      Afficher (i, "est un nombre
parfait")

    Afficher (saut de ligne)
  Fin si
Fin pour
Fin
```