Chapitre 6 – Les fonctions et les procédures

I. Introduction

Certains problèmes conduisent à des programmes longs, difficiles à écrire et à comprendre. On les découpe en des parties appelées **sous-programmes** ou **modules.**

Les **fonctions** et les **procédures** sont des modules (groupe d'instructions) indépendants désignés par un nom. Elles ont plusieurs **intérêts** :

- Permettent de **"factoriser" les programmes**, c.-à-d. de mettre en commun les parties qui se répètent.
- Permettent une **structuration** et une **meilleure lisibilité** des programmes.
- **Facilitent la maintenance** du code (il suffit de modifier une seule fois).
- Ces procédures et fonctions peuvent éventuellement être **réutilisées** dans d'autres programmes.

II. Les fonctions

Le **rôle** d'une fonction en programmation est similaire à celui d'une fonction en mathématique : elle **retourne un résultat à partir des valeurs des paramètres**

Une fonction est un mini-programme qu'on déclare dans la partie réservée aux variables, ce afin de pouvoir utiliser les variables globales. Étant donné qu'il s'agit d'un bloc à part entière, elle possèdera éventuellement un en-tête, une série de traitements, et une gestion des résultats tout comme l'algorithme qui la contient. En outre, une fonction

peut également recevoir des arguments qui lui seront alors passés en paramètres.

La syntaxe de déclaration d'une fonction :

FONCTION nom_de_la_fonction **[(liste des paramètres : type)]** : **type_fonction**

Variable : liste des variables

DEBUT {corps de la fonction}

{la liste ne doit pas être vide}

retourne….

FINFONCTION

Remarques :

- Pour le choix d'un nom de fonction il faut respecter les mêmes règles que celles pour les noms de variables
- **Type_fonction** est le type du résultat retourné
- L'instruction **retourne** sert à retourner la valeur du résultat
- La liste des paramètres est facultative. Mais quand elle existe, ces paramètres sont déclarés de la même façon qu'on déclare une série de variables de différents types.
- Les variables déclarées à l'intérieur de la fonction sont inutilisables à l'extérieur du bloc.

Exemple :

```
Algorithme Exemple_1
Variables x, y, Res : entier
    FONCTION Somme(a : entier, b : entier) : entier
        Variable s : entier
    Début
        s ← a + b
        retourne s
    FinFonction
Début
    Ecrire("Donner la valeur de x :")
    Lire(x)
    Ecrire("Donner la valeur de y :")
    Lire(y)
    Res = Somme(x , y)
    Ecrire(x,"+",y,"=",res)
    Res = Somme(2, 4)
    Ecrire("2+ 4=",res)
    Res = Somme(6,7)
    Ecrire("6+ 7=",res)
Fin
```

Exercices d'application

Exercice 1 :

Ecrire une fonction qui retourne la somme de deux nombres passé en paramètre.

Exercice 2 :

Ecrire une fonction qui retourne le nombre maximal de deux nombres passés en paramètre.

Exercice 3 :

Ecrire une fonction booléenne qui permet de retourner si un nombre est pair ou non.

Exercice 4 :

Ecrire une fonction booléenne qui permet de retourner si un nombre est premier ou non.

Exercice 5

Ecrire une fonction qui retourne le factoriel d'un nombre passé en paramètre.

Exercice 6 :

Ecrire une fonction qui retourne la moyenne de trois nombre passé en paramètre.

Exercice 7 :

Écrire la fonction NCHIFFRES du type entier qui obtient une valeur entière N (positive) du type long entier comme paramètre et qui fournit le nombre de chiffres de N comme résultat.

Exemple:

Introduire un nombre entier : 6457392

Le nombre 6457392 a 7 chiffres.

Exercice 8 :

Ecrire une fonction qui reçoit en arguments 2 nombres réels et un caractère et qui fournit un résultat correspondant à l'une des 4 opérations appliquées à ses deux premiers arguments, en fonction de la valeur de ce dernier, à savoir : addition pour le caractère +, soustraction pour le caractère -, multiplication pour * et division pour / (tout autre caractère sera interprété comme une addition). On ne tiendra pas compte des risques de la division par 0.

III. Les procédures

Dans certains cas, on peut avoir besoin de répéter une tache dans plusieurs endroits du programme, mais que dans cette tache on ne calcule pas de résultats ou qu'on calcule plusieurs résultats à la fois

Dans ces cas on ne peut pas utiliser une fonction, on utilise une **procédure**

Une **procédure** est un sous-programme semblable à une fonction mais qui **ne retourne rien**

Une procédure s'écrit en dehors du programme principal sous la forme :

```
Procédure nom_procédure (paramètres et leurs types)
    Variable : Liste des variable
    Instructions constituant le corps de la procédure
FinProcédure
```

Remarque : une procédure peut ne pas avoir de paramètres

L'appel d'une procédure

L'appel d'une procédure, se fait dans le programme principale ou dans une autre procédure par une instruction indiquant le nom de la procédure :

```
Procédure exemple_proc (…)
    …
FinProcédure
Algorithme exepmleAppelProcédure
Début
    exemple_proc (…)
    …
Fin
```

Remarque : contrairement à l'appel d'une fonction, on ne peut pas affecter la procédure appelée ou l'utiliser dans une expression. L'appel d'une procédure est une instruction autonome

Les paramètres d'une procédure

Les paramètres servent à échanger des données entre le programme principale (ou la procédure appelante) et la procédure appelée

Les paramètres placés dans la déclaration d'une procédure sont appelés **paramètres formels**. Ces paramètres peuvent prendre toutes les valeurs possibles mais ils sont abstraits (n'existent pas réellement)

Les paramètres placés dans l'appel d'une procédure sont appelés **paramètres effectifs**. Ils contiennent les valeurs pour effectuer le traitement

Le nombre de paramètres effectifs doit être égal au nombre de paramètres formels. L'ordre et le type des paramètres doivent correspondre

Transmission des paramètres

Il existe deux modes de transmission de paramètres dans les langages de programmation :

- **La transmission par valeur** : les valeurs des paramètres effectifs sont affectées aux paramètres formels correspondants au moment de l'appel de la procédure. Dans ce mode le paramètre effectif ne subit aucune modification
- **La transmission par adresse (ou par référence)** : les adresses des paramètres effectifs sont transmises à la procédure appelante. Dans ce mode, le paramètre effectif subit les mêmes modifications que le paramètre formel lors de l'exécution de la procédure

Remarque : le paramètre effectif doit être une variable (et non une valeur) lorsqu'il s'agit d'une transmission par adresse

En algorithme, on va préciser explicitement le mode de transmission dans la déclaration de la procédure

Exemple 1 :

Procédure incrementer1 (**x** : entier **par valeur, y** : entier **par adresse**)

x ← x+1

y ← y+1

FinProcédure

Algorithme Test_incrementer1

variables n, m : entier

Début

n ← 3

m ← 3

incrementer1(n, m) **résultat :**

écrire (" n= ", n, " et m= ", m) **n=3 et m=4**

Fin

Remarque : l'instruction x ← x+1 n'a pas de sens avec un passage par valeur

Exemple 2 :

Procédure qui calcule la somme et le produit de deux entiers :

Procédure SommeProduit (x,y: entier **par valeur,** som, prod : entier **par adresse**)

som ← x+y

prod ← x*y

FinProcédure

Procédure qui échange le contenu de deux variabales :

Procédure Echange (**x** : réel **par adresse, y** : réel **par adresse**)

variables z : réel

z ← x

x ← y

y ← z

FinProcédure

Les variables locales et globales

On peut manipuler 2 types de variables dans un module (procédure ou fonction) : des **variables locales** et des **variables globales**. Elles se distinguent par ce qu'on appelle leur **portée** (leur "champ de définition", leur "durée de vie")

Une **variable locale** n'est connue qu'à l'intérieur du module ou elle a été définie. Elle est créée à l'appel du module et détruite à la fin de son exécution

Une **variable globale** est connue par l'ensemble des modules et le programme principale. Elle est définie durant toute l'application et peut être utilisée et modifiée par les différents modules du programme

Exercices d'application

Exercice 1 :

Écrire :

- Une procédure, nommée f1, se contentant d'afficher bonjour (elle ne possèdera aucun argument).
- Une procédure, nommée f2, qui affiche bonjour un nombre de fois égale à la valeur reçue en argument (entier).
- Une fonction, nommée, f3, qui fait la même chose que f2, mais qui, de plus, renvoie la valeur (entier) 0.

Exercice 2 :

Écrire un programme dans lequel une procédure prend en paramètres un prénom et un âge. La procédure affichera ensuite un message disant 'BONJOUR', PRENOM, puis indiquera s'il s'agit d'un enfant (-20), d'un adulte (-50), ou d'une personne âgée.

Exercice 3 :

Écrire une procédure qui prend en paramètres 2 entiers p et n rentrés par l'utilisateur dans le bloc principal, et qui calcule ensuite la valeur p^n. p devra changer de valeur pour prendre cette nouvelle valeur.

Exercice 4 :

Ecrire une procédure qui affiche le PGCD de deux nombres passés en paramètres.

Exercice 5 :

Ecrire une procédure qui donne la solution d'une équation de 2^{éme} degré.