LES FONCTIONS ET LES PROCEDURES

I- INTRODUCTION

Lorsque l'on progresse dans la conception d'un algorithme, ce dernier peut prendre une taille et une complexité croissante. De même des séquences d'instructions peuvent se répéter à plusieurs endroits. La solution consiste alors à découper l'algorithme en plusieurs parties plus petites. Ces parties sont appelées des **sous-programmes (fonctions** et **procédures)**.

Il existe deux sortes de sous-algorithmes :

- **Les fonctions**
- **Les procédures**

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

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

II- LES FONCTIONS

Une Fonction est une partie d'un algorithme (ou un sous-programme) ; qui reçoit des valeurs suite un appel.

SYNTAXE :

Fonction nom_Fonction (parm1, parm2, …) : Type_résultat

Début

Instruction 1

Instruction 2

……

Instruction n

Retourner résultat

Fin

REMARQUES :

- Pour le choix d'un nom de fonction il faut respecter les mêmes règles que celles pour les noms de variables
- Le type de fonction est le type du résultat retourné (Entier, réel, booléen, etc.)
- L'instruction retourner sert à retourner la valeur du résultat
- Une fonction peut avoir de 0 à N paramètres
- Param1, param2, …, sont appelés paramètres(Arguments) : ce sont des variables qui permettent à la fonction de communiquer avec l'extérieur.
- Ces paramètres déclarés lors de la définition de la fonction sont appelés paramètres formels.

EXEMPLE : ----------

Ecrire une fonction nommée **maximum** qui calcule et renvoie le plus grand de deux entiers A et B.

```
Fonction maximum (A : entier, B : entier) : entier
Début
    Si A>B alors
    Max ← A sinon
    Max ← B Fin Si
    Retourner max
Fin
```

L'APPEL D'UNE FONCTION :

Pour exécuter une fonction, il suffit de faire **appel à elle en écrivant son nom suivi des paramètres effectifs dans le programme principal**.

Le résultat d'une fonction étant une valeur, devra être affecté à une variable où être utilisé dans une expression, calcul, affichage, test, …etc.

EXEMPLE : ----------

Appel de la fonction sommeCarre qui renvoie la somme carré deux nombres.

```
Algorithme : Somme_deux_carre
    //Déclaration de la fonction sommeCarre()
    Fonction sommeCarre (x : réel, y : réel):réel
    Début:
    Resultat ← x*x + y*y
    Retourner Resultat
```

```
        Fin
//Algorithme principal
Début
Ecrire("saisir une valeur de X ") Lire(x)
Ecrire("saisir une valeur de Y ") Lire(x)
//Appel de la fonction sommeCarre()
Rst← sommeCarre(x, y)
Ecrire("La somme carré est :", Rst)
Fin
```

EXEMPLE 2 : ----------

Appel de la fonction maximum qui renvoie le maximum deux nombres.

```
Algorithme : maximum_deux_nombres
        //Déclaration de la fonction maximum()
        Fonction maximum(A : entier, B : entier) : entier
         Début
                Si A>B alors
                 max←A
                sinon
                max←B
        Fin Si
        Retourne max
        Fin
//Algorithme principal
Début:
Ecrire("saisir une valeur de A ") Lire(A)
Ecrire("saisir une valeur de B ") Lire(B)
//Appel de la fonction maximum()
max← maximum(A, B)
Ecrire("Le maximum est: ",max)
```

Fin

NB : Lors de l'appel de la fonction maximum (A, B) les deux paramètres formels A et B sont remplacés par les valeurs A et B (Paramètres effectifs) saisie par l'utilisateur.

III- LES PROCEDURES

Une procédure est une série d'instructions regroupés sous un nom, qui permet d'effectuer des actions par un simple appel de la procédure dans un algorithme ou dans un autre sous-algorithme.

SYNTAXE :

```
Procédure nom_Procédure (parm1, parm2,…)
Début
    Instruction 1
    Instruction 2
    ……
    Instruction n
Fin Procédure
```

APPEL DE LA PROCEDURE :

Pour déclencher l'exécution d'une procédure dans un algorithme (programme), il suffit de l'appeler.

L'appel de procédure s'écrit en mettant le nom de la procédure, puis la liste des paramètres, séparés par des virgules.

A l'appel d'une procédure, l'algorithme (programme) interrompt son déroulement normal, exécute les instructions de la procédure, puis retourne au programme appelant et exécute l'instruction suivante.

SYNTAXE :

```
nom_Prcocedure (Pram1, Param2, …)
```

EXEMPLE:

Appel de la procédure permuter qui prend en paramètres deux nombres A et B de type entier puis affiche le résultat de permutation.

Algorithme : permutation

```
//Déclaration de la procédure permuter()
Procédure permuter (@A : entier, @B : entier)
```

```
Début
        C←A
        A←B
        B←C
Ecrire("La nouvelle valeur de A :", A)
Ecrire("La nouvelle valeur de B :", B)
Fin
//Algorithme principal
Début
Ecrire("saisir une valeur de A ") Lire(A)
Ecrire("saisir une valeur de B ") Lire(B)
//Appel de la procédure permuter()
Permuter(A, B)
Fin
```

REMARQUES :

- 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.

IV- Paramètres d'une fonction ou d'une procédure :

Les paramètres servent à échanger des données entre l'algorithme principal et la procédure ou la fonction appelée.

Lors de l'appel d'une fonction ou d'une procédure, deux formes de paramètres entrent en jeu : les paramètres formels et les paramètres effectifs.

Les paramètres formels :

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

Les paramètres effectifs :

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

Remarque :

Les paramètres formels et les paramètres effectifs doivent correspondre en nombre, en type et en ordre. Les noms peuvent se différer

Il existe deux principaux types de passages de paramètres qui permettent des usages différents :

1- **Passage de valeur :**

Dans ce type de passage, le paramètre formel reçoit uniquement une copie de la valeur du paramètre effectif. La valeur du paramètre effectif ne sera jamais modifiée.

2- **Passage par référence ou par adresse :**

Dans ce type de passage, la procédure utilise l'adresse du paramètre effectif. Lorsqu'on utilise l'adresse du paramètre, on accède directement à son contenu. La valeur de la variable effective sera donc modifiée.

Les paramètres passés par adresse sont précédés de **@**.

Portées des variables :

On peut manipuler deux types de variables dans une 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")

Les variables locales :

Une variable déclarée à l'intérieur d'une procédure/fonction est dite locale. Elle n'est accessible qu'à la procédure au sein de laquelle elle définit, les autres procédures/fonctions n'y ont pas accès. La durée de vie d'une variable locale est limitée à la durée d'exécution de la procédure/fonction.

Les variables globales :

Une variable déclarée dans la partie déclaration de l'algorithme principale est appelée variable globale. Elle est accessible de n'importe où dans l'algorithme, même depuis les procédures et les fonctions. Elle existe pendant toute la durée de vie d'algorithme (programme).