Résumé sur les sous-programmes

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.

Un algorithme écrit d'un seul tenant devient difficile à comprendre et à gérer dès qu'il dépasse deux pages. La solution consiste alors à découper l'algorithme en plusieurs parties plus petites. Ces parties sont appelées des sous-algorithmes.

Le sous-algorithme est écrit séparément du corps de l'algorithme principal et sera appelé par celui-ci quand ceci sera nécessaire.
Il existe deux sortes de sous-algorithmes : les procédures et les fonctions.

II. Les procédures

Une procédure est une série d'instructions regroupées 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.
Une procédure renvoie plusieurs valeurs (pas une) ou aucune valeur.

II.1. Déclaration d'une procédure

Syntaxe :

```
Procédure nom_proc (liste de paramètres)
Variable identificateurs : type
Début
    Instruction(s)
Fin Proc
```

II.2. L'appel d'une procédure

Pour déclencher l'exécution d'une procédure dans un 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, le programme interrompt son déroulement normal, exécute les instructions de la procédure, puis retourne au programme appelant et exécute l'instruction suivante.

Syntaxe :

Nom_proc(liste de paramètres)

III. Les fonctions

Les fonctions sont des sous algorithmes admettent des paramètres et retournant un seul résultat (une seule valeur) de type simple qui peut apparaitre dans une expression, dans une comparaison, à la droite d'une affectation, etc.

III.1. Déclaration d'une fonction

Syntaxe :

```
Fonction nom_Fonct(liste de paramètres) : type
Variables identificateurs : type
Début
    Instruction(s)
    Retourner Expression
FinFonct
```

III.2. L'appel d'une fonction

Pour exécuter une fonction, il suffit de faire appel à elle en écrivant son nom suivi des paramètres effectifs. C'est la même syntaxe qu'une procédure.

A la différence d'une procédure, la fonction retourne une valeur. L'appel d'une fonction pourra donc être utilisé dans une instruction (affichage, affectation,…) qui utilise sa valeur.

Syntaxe :

Nom_Fonction(liste de paramètres)

Quelques rappels utiles :

Rappel 1 :

Les *paramètres formels* sont les paramètres utilisés dans la déclaration des procédures et fonctions. Par contre, les *paramètres effectifs* sont les paramètres utilisés lors de l'appel aux procédures et fonctions. (Les paramètres formels sont séparés par des points-virgules).

Rappel 2 :

Transmission des paramètres par valeur

Pour la transmission de paramètres par valeur, il suffit de déclarer les paramètres formels sans utiliser le mot clé ***VAR***.

Transmission des paramètres par variables

On utilise la transmission par variable lorsqu'on veut que la variable du paramètre formel dans la fonction ou procédure affecte le paramètre effectif correspondant au niveau du programme appelant. Dans ce cas, le paramètre formel doit être précédé du mot clé ***VAR***. Le paramètre effectif correspondant *doit être, obligatoirement, une variable*.

En d'autres termes, on utilise une transmission par variable lorsqu'on veut recueillir le résultat dans cette variable. *(Variable résultat)*.

Rappel 3 :

Transmission des paramètres par fonction

Dans ce cas, un nom de fonction est utilisé comme paramètre. On doit alors indiquer que le paramètre formel correspondant est une fonction le précédant du mot clé ***FUNCTION***.

Transmission des paramètres par procédure

Dans ce cas, un nom de procédure est utilisé comme paramètre. On doit alors indiquer que le paramètre formel correspondant est une procédure en le précédant du mot clé ***PROCÉDURE***.

Remarques :

Afin d'assurer la transmission correcte des paramètres effectifs vers les paramètres formels, il est nécessaire que les paramètres effectifs correspondent en nombre et en type aux paramètres formels correspondants. Autrement dit, le nombre de paramètres effectifs doit être égal au nombre de paramètres formels, et les types des paramètres effectifs doivent correspondre ou être compatibles avec les types des paramètres formels.