Chapitre 7 : La Programmation Modulaire

Introduction

Généralement dans un programme, nous distinguons plusieurs parties. Pour des besoins de lisibilité, d'optimisation et de réutilisabilité de code, il est recommandé de considérer chaque partie et d'en faire un module encore appelé sous-programme. Dans un programme, les variables sont classées en deux catégories :

- Les variables locales : elles sont déclarées dans un module et leur visibilité est limitée à ce module. Elles sont composées des variables d'en-têtes et intermédiaires.
- Les variables globales : Elles sont déclarées au niveau du programme principal et leur visibilité s'étend dans tout le programme.

Un module n'est connu que s'il est appelé. Dans ce cas, on parle d'appel de module. L'appelant et l'appelé vont s'échanger en utilisant leurs arguments. Cet échange utilise soit le mode de transmission par copie de valeurs ou soit le mode de transmission par référence. Le mode de transmission à utiliser est déterminé par le type de paramètre des arguments de l'appelé.

Un module est défini soit de façon itérative, soit de façon récursive en utilisant soit une procédure soit une fonction

I. La procédure

1. Définition

Une procédure est un module qui peut recevoir des arguments donnés, mais également elle peut en avoir des arguments résultats.

2. Prototype de la procédure

```
Procedure nomProcedure(Types(s) de paramètre argument(s): type(s))
//Variable(s) intermediaire(s)
DEBUT
        //Corps de la procedure
FIN
```

NB : type de paramètre	Donnee(D)=Entree(E)=Input(I) Résultat(R)=Sortie(S)=Output(O) D/R=E/S=I/O

- Le type de paramètre Donnée permet de spécifier les arguments indispensables à l'exécution du module
- Le type de paramètre Résultat permet d'énoncer les arguments qui seront créés après l'exécution du module

- Le type de paramètre Donnée/Résultat permet de marquer les arguments utiles à l'exécution du module mais leurs valeurs peuvent être modifiées après exécution de la procédure

Exercices d'application :

1) Ecrire une procédure qui reçoit 2 entiers puis détermine leur somme.
2) Ecrire une procédure qui reçoit un tableau d'entiers puis décale ses valeurs de façon cyclique d'une position à droite. La taille du tableau est égale à 250
3) Ecrire une procédure qui reçoit un entier puis affiche vrai si l'entier est un nombre premier sinon faux dans le cas contraire

II. La fonction

1. Définition

Une fonction est un module qui peut avoir des arguments Donnée mais elle a une obligation de retourner un Résultat.

2. Prototype de la fonction

Fonction nomFonction (Donnee(s) Argument(s) : type(s)) : typeRetour

```
//Variable(s) intermediaire(s)
DEBUT
        //Corps de la fonction
        Retourner  expression
FIN
```

NB : Le type de valeur retourné par la fonction doit être compatible au type de l'expression retournée par la fonction

Applications :

1) Ecrire une fonction qui reçoit un entier puis détermine son factoriel.
2) Ecrire une fonction qui reçoit un entier puis détermine le résultat de la série harmonique définie comme suit : $\sum_1^n(\frac{1}{i})$

III. Les modes de transmission

Ces modes de transmission sont utilisés pour établir une communication entre les arguments de l'appelant et ceux de l'appelé. Ils sont au nombre de 2 :

- Le mode de transmission par copie de valeur : Il permet aux arguments de l'appelant de donner à ceux de l'appelé une copie de leur valeur mémoire. L'appelé manipule une copie des valeurs de l'appelant. Ce mode de transmission est utilisé lorsque le type de paramètre des arguments est D (= E ou I)

- Le mode de transmission par référence, également appelé mode transmission par variable ou par adresse : Il permet aux arguments de l'appelant de donner à ceux de l'appelé leur adresse mémoire. Dans ce cas, ces arguments manipulent les mêmes adresses donc les mêmes valeurs. Il est utilisé lorsque le type de paramètre des arguments de l'appelé est R ou D/R

APPLICATION :

Ecrire un programme contenant les modules suivants :

- Un module de saisie d'un entier
- Un module qui reçoit 2 entiers et les affiche
- Un module qui reçoit 2 entiers et échange leur contenu

Il faut s'arranger à ce que le programme permette de saisir 2 valeurs, de les afficher, d'échanger leur contenu et de les afficher après échange.

IV. La récursivité

La récursivité est un concept mathématique qui peut être programmé en utilisant des modules récursifs. Un module est récursif s'il s'auto-réfère dans son exécution. L'auto-appellation peut se faire de façon directe ou transitive.

Exemple de formules récursives :

1) factoriel $N! = N * (N - 1)!$

2) Suite de Fibonacci $U_0=U_1=1$; $u_n = u_{n-1} + u_{n-2}, n > 1$

3) Fonction Puissance

NB : Un module récursif n'a pas besoin de boucles pour s'exécuter. La répétition des traitements est liée à l'auto-appellation du module qui va s'arrêter dès que la condition minimale sera atteinte.

Application :

1. Ecrire un module récursif qui reçoit un entier puis détermine son factoriel
2. Ecrire un module récursif qui reçoit le terme N d'une suite puis détermine le résultat en utilisant en utilisant la suite de Fibonacci.