Les sous-programmes

Il arrive fréquemment qu'une même suite d'instruction doive être exécutée en plusieurs points du programme. Un sous-programme permet d'associer un nom à une suite d'instruction et d'utiliser ce nom comme abréviation chaque fois que la suite apparaît dans le programme.

Pour alléger l'écriture d'un algorithme, il est conseillé de rédiger des sous-programmes qui sont appelés dans le corps du programme.

La modularité des programmes ainsi obtenue assure :

- une meilleure qualité de programmation
- un code plus court et par suite une diminution du risque d'erreurs de syntaxe,
- une mise au point aisée et rapide des programmes,
- une facilité de maintenance accrue,
- une facilité d'intégration de modules à d'autres programmes,
- une centralisation de la résolution d'un problème autour d'un programme principal.

Cette technique de décomposition initiale d'un problème en plusieurs problèmes distincts, plus simples, permet surtout à l'utilisateur de concevoir et de construire ses propres bibliothèques de sous-programmes.

En algorithmique il existe deux types de sous-programmes :

Les fonctions

Les procédures

Un sous-programme est obligatoirement caractérisé par un nom (un identifiant) unique.

Lorsqu'un sous programme a été explicité (on a donnée l'algorithme), son nom devient une nouvelle instruction, qui peut être utilisé dans d'autres (sous-)programmes
Le (sous-)programme qui utilise un sous-programme est appelé (sous-)programme appelant.
Lorsqu'un sous-programme A appelle un sous-programme B :
Il faut, lorsque l'on définit l'algorithme de B, que les informations qu'on lui donne soit identifiées pour pouvoir être utilisées on parle de paramètre formel.
Il faut que A donne des informations à B lors de l'appel on parle de paramètre effectif (aussi appel´e argument).
Un paramètre formel d'un sous-programme est une variable locale particulière Il a donc un type.
Un paramètre effectif est une variable ou constante. Le paramètre formel et le paramètre effectif sont associés lors de l'appel du sous-programme. Cette association est effectuée suivant l'ordre de ces paramètres
Les paramètres formel et effectif doivent donc être de même type.

Passage de Paramètres

Il existe trois types d'association (que l'on nomme passage de paramètre) entre le paramètre formel et le paramètre effectif du (sous-)programme appelant :

- Le passage de paramètre en entrée
- Le passage de paramètre en sortie
- Le passage de paramètre en entrée/sortie

Les Fonctions

Les fonctions sont des sous-programmes admettant des paramètres formels et retournant un seul résultat (comme les fonctions mathématiques y=f(x,y,. . .)) les paramètres formels sont en nombre fixe (supérieur ou égal à zéro) une fonction possède un seul type, qui est le type de la valeur retournée le passage de paramètre est uniquement en entrée : c'est pour cela qu'il n'est pas précisé lors de ~~l'appel~~, on peut donc utiliser comme paramètre des variables, des constantes mais aussi des résultats de fonction . La valeur de retour est spécifiée par l'instruction retourner
Généralement le nom d'une fonction est soit un nom (par exemple minimum)

On déclare une fonction de la façon suivante :
fonction *nom de la fonction* **(***paramètre(s) de la fonction***) :** *type de la valeur retournée*
    Déclaration *variable(s) locale(s)*
debut
    instructions de la fonction avec au moins une fois l'instruction **retourner**
fin

On utilise une fonction en précisant son nom suivi des paramètres effectifs entre parenthèses

- Les parenthèses sont toujours présentes même lorsqu'il n'y a pas de paramètre effectif

Exemple

```
fonction minimum2 (a,b : Entier) : Entier
debut
    si a ≥ b alors
        retourner b
    sinon
        retourner a
    finsi
fin
fonction minimum3 (a,b,c : Entier) : Entier
debut
    retourner minimum2(a,minimum2(b,c))
fin
```

Les Procédures

Les procédures sont des sous-programmes qui ne retournent **aucun** résultat

Par contre elles admettent des paramètres avec des passages :

- en entrée, préfixés par **Entrée** (ou **E**)
- en sortie, préfixés par **Sortie** (ou **S**)
- en entrée/sortie, préfixés par **Entrée/Sortie** (ou **E/S**)

On déclare une procédure de la façon suivante :

procédure *nom de la procédure* **(** **E** *paramètre(s) en entrée ;* **S** *paramètre(s) en sortie ;* **E/S** *paramètre(s) en entrée/sortie* **)**

    Déclaration *variable(s) locale(s)*

debut

    instructions de la procédure

fin

On appelle une procédure comme une fonction, en indiquant son nom suivi des paramètres effectifs entre parenthèses

Exemple

```
procédure calculerMinMax3 ( E a,b,c : Entier ; S m,M : Entier )
debut
    m ← minimum3(a,b,c)
    M ← maximum3(a,b,c)
fin
```

Algorithme

PROCEDURE NomProcédure(paramètre1 : type) ;

FONCTION NomFonction(Param1, Param2 : type) : Type ;

DEBUT

Corps de l'algorithme

NomProcédure(variable) %la procédure est appelée et le contenu de variable est passé en Paramètres%

ResultatFonction ← NomFonction(variable2) %le contenu de variable2 est passé en paramètre de la fonction%

FIN

PROCEDURE NomProcédure(paramètre1 : type)

Actions internes à la procédure

FIN PROCEDURE

FONCTION NomFonction(Param1, Param2 : type) : Type

Actions internes à la fonction

FIN FONCTION

Remarque

- Une procédure exécute un traitement mais ne renvoie pas de valeur
- Une fonction renvoie des valeurs pouvant être exploitées dans le corps du programme

Exercices

Tests

Un sous-programme est obligatoirement caractérisé par :

- un nom unique
- un chiffre
- un type

Une fonction renvoie une ou plusieurs valeurs

- faux
- vrai

Une procédure renvoie une seule valeur

- faux
- vrai