Procédures, fonctions et passage de paramètres

1. Les objectifs

Ces nouveaux mécanismes permettent au programmeur de traiter un problème sans se soucier, dans un premier temps, du règlement dans le détail des sous-problèmes. Il s'agit d'outils très similaires aux fonctions mathématiques.

En programmation, une procédure ou une fonction représente un algorithme opérant sur des arguments ou paramètres formels (valeurs fictives en quelque sorte). L'exécution de cet algorithme se produit à l'appel de la procédure ou de la fonction. Les données de cet algorithme appelé sont alors les paramètres effectifs (provenant de l'algorithme appelant).

L'intérêt méthodologique de ces mécanismes consiste essentiellement, dans une première étape, à convertir les sous-problèmes en procédures (ou en fonctions) et à traiter le problème général (ou principal) comme si ces sous-problèmes étaient résolus. Dans une seconde étape, ces procédures ou fonctions sont décrites. Un autre intérêt réside dans le fait que si le même sous-problème doit être résolu plusieurs fois avec des paramètres effectifs différents, l'emploi d'une procédure ou d'une fonction permet une meilleure lisibilité de l'algorithme.

2. Les procédures

Débutons l'étude de ce mécanisme par la syntaxe de la déclaration :

```
Procédure NOM_PROCEDURE([Type1 IDFO1, ..., Typei IDFOi])
        Corps de la procédure
Fin_procédure
```

Dans cette déclaration :

- NOM_PROCEDURE est un nom librement choisi par le programmeur.
- IDFOi est le nom du ième paramètre formel de la procédure.
- ***Typei*** est le type du ième paramètre formel de la procédure.
- La procédure ne rend pas de valeurs à moins qu'elle n'intervienne sur des variables de portée (accessibilité) globale.
- ***Procédure*** et ***Fin_procédure*** sont des mots réservés.

L'appel d'une procédure depuis une autre procédure ou depuis l'algorithme principal se fait en respectant le schéma suivant :

```
NOM_PROCEDURE(IDEF1, IDEFi, ..., IDEFn)
```

avec :

- NOM_PROCEDURE qui est un nom librement choisi par le programmeur,
- IDEFi qui est le nom du ième paramètre effectif.

3. Exercice n°16 : Appel d'une procédure avec passage de paramètres

Sujet

Appeler une procédure affichant le double des valeurs effectives passées en paramètres (les deux valeurs passées en paramètres seront à saisir au clavier).

Corrigé

```
Début

    Co Utilisation d'une procédure affichant le double d'un couplet de valeurs Fco

    Co Procédure DOUBLE Fco
    Procédure DOUBLE(Ent X, Ent Y)

        X <- X * 2
        Y <- Y * 2
        Ecrire("Le couplet doublé est (", X, ",", Y, ")")

    Fin_procédure

    Co Traitement appelant Fco
    Ent A, B, C, D
    Ecrire("Programme doublant les valeurs de couplets")
    Ecrire("Valeur 1 du 1er couplet : ")
    A <- Lire
    Ecrire("Valeur 2 du 1er couplet : ")
    B <- Lire
    DOUBLE(A, B)
    Ecrire("Valeur 1 du 2ème couplet : ")
    C <- Lire

    Ecrire("Valeur 2 du 2ème couplet : ")
    D <- Lire
    DOUBLE(C, D)

Fin
```

Dans le corrigé présenté en réalité deux couplets de nombres sont saisis au clavier (A et B puis C et D) et ceci pour montrer la réutilisabilité de la procédure DOUBLE.

Les variables A et B (puis C et D) sont qualifiés de paramètres effectifs. Lors de l'appel de la procédure DOUBLE, le paramètre effectif A vient s'imputer dans le paramètre formel X et le paramètre effectif B vient s'imputer dans le paramètre formel Y. La procédure DOUBLE effectue ensuite un doublement des valeurs passées en paramètres et affiche les résultats. Il faut bien noter qu'il n'y a pas de retour de valeurs à destination de la procédure appelante. Les valeurs A et B conserveront donc leurs valeurs initiales. Il n'est pas possible non plus que la procédure DOUBLE agisse directement sur les variables A et B (ou encore C et D).

Dans un passage de paramètres entre une procédure appelante (ou un algorithme principal) et une procédure appelée quelques règles sont à considérer :

- Les paramètres effectifs (s'ils sont multiples) doivent être séparés par des virgules au niveau de l'appel.
- Le(s) paramètre(s) formel(s) doit (doivent) être déclaré(s) dans le jeu de parenthèses suivant le nom de la fonction (ou de la procédure) appelée.
- Les paramètres formels (s'ils sont multiples) doivent être séparés par des virgules.
- Chaque paramètre effectif correspond à un paramètre formel et les deux doivent être du même type.

Il est possible de passer des valeurs de types différents entre une procédure et une autre procédure (ou une fonction).

4. Les fonctions

La différence essentielle entre les procédures vues précédemment et les fonctions est que dans le cas des fonctions une valeur est renvoyée à destination de la procédure appelante (ou de l'algorithme principal).

La syntaxe de déclaration est donc assez proche de celle d'une procédure :

```
Fonction NOM_FONCTION([Type1 IDFO1, ..., Typei IDFOi]) Type_résultat
        Corps de la fonction (contenant Retourne expression)
Fin_fonction
```

Dans cette déclaration :

- L'appel de la fonction fournit une valeur du type ***Type_résultat*** indiqué (***Ent***, ***Réel***, ***Bool***, ***Car***).
- NOM_FONCTION est un nom librement choisi par le programmeur.
- IDFOi est le nom du ième paramètre formel de la fonction.
- ***Typei*** est le type du ième paramètre formel de la fonction.
- La fonction doit rendre une expression (de type ***Type_résultat***) sous la forme ***Retourne*** expression.
- ***Fonction***, ***Résultat***, ***Retourne*** et ***Fin_fonction*** sont des mots réservés.

L'appel d'une fonction depuis une autre procédure ou depuis l'algorithme principal se fait en respectant le schéma suivant :

```
RESULTAT = NOM_FONCTION (IDEF1, IDEFi, ..., IDEFn)
```

avec :

- NOM_FONCTION qui est un nom librement choisi par le programmeur,
- IDEFi qui est le nom du ième paramètre effectif,
- RESULTAT qui est le nom de la variable récupérant la réponse fournie par la fonction.

5. Exercice n°17 : Appel d'une fonction avec passage de paramètres

Sujet

Utilisation d'une fonction de calcul du maximum de deux valeurs saisies au clavier et passées en paramètres

Corrigé

```
Début
        Co Utilisation d'une fonction de calcul du maximum de deux valeurs
        saisies au clavier et passées en paramètres Fco
```

```
    Co Fonction de calcul du maximum Fco
    Fonction MAXI(Ent X, Ent Y) Ent
        Si X > Y
        Alors
            Retourne X
        Sinon
            Retourne Y
        Finsi
    Fin_fonction

    Co Traitement appelant Fco
    Ent A, B, C, D, E
    Ecrire("Calcul du maximum de 2 nombres")
    Ecrire("Valeur 1 : ")
    A <- Lire
    Ecrire("Valeur 2 : ")
    B <- Lire
    Ecrire("Le + grand est ", MAXI(A, B))
    Ecrire("Valeur 3 : ")
    C <- Lire
    Ecrire("Valeur 4 : ")
    D <- Lire
    E <- MAXI(C, D)
    Ecrire("Le + grand est ", E)
Fin
```

Dans ce corrigé, une même fonction MAXI est appelée deux fois avec un jeu de paramètres effectifs différent (A et B puis C et D).

Les règles de passage de paramètres entre une procédure appelante (ou un algorithme principal) et une fonction appelée sont identiques à celles vues précédemment dans le cadre d'un appel de procédure.