TP Informatique 24 - Initiation aux procédures

Nous avons déjà vu la notion de **fonction** en Pascal. C'était un mini programme qui renvoyait un résultat "chiffré" qu'on pouvait ensuite appliquer en tout endroit du programme.
L'inconvénient avec une fonction est qu'elle ne fait que calculer : on ne peut rien demander à l'utilisation, on ne peut rien afficher ...
Pour contourner ce problème, on introduit la notion de **procédure**. De même que pour une fonction, une procédure est un sous-programme (qu'on placera avant le `BEGIN` du programme) qui définit une liste d'instructions utilisable et ré-utilisable en tout endroit du programme.

Procédures sans paramètre

Une procédure peut être, dans le cas le plus simple, juste une liste d'instructions :
Une procédure porte juste un nom, on l'écrit avant le programme (avant le `BEGIN`) eton l'appelle autant de fois qu'on veut dans le programme.

Exercice 24.1

Que fait la procédure suivante :

```
PROGRAM exemple_procedure ;
    VAR k,n,p : INTEGER ;
PROCEDURE action ;
    BEGIN
    p := k ;
    k := n ;
    n := p ;
    END ;
BEGIN
k := 3 ;
n := 4 ;
WRITELN(k,' ',n) ;
action ;
WRITELN(k,' ',n) ;
action ;
WRITELN(k,' ',n) ;
READLN ;
END.
```

1 Appel de procédures depuis une autre procédure

On peut appeler une procédure P_1 à l'intérieure d'une procédure P_2.
Pour cela, il suffit de déclarer la procédure P_1 avant la procédure P_2.

Exercice 24.2

Que fait le programme suivant ?

```
PROGRAM procedure_dans_une_procedure ;
    VAR k,n,p : INTEGER ;
PROCEDURE affiche ;
    BEGIN
    WRITELN(k,' ',n) ;
    END ;
PROCEDURE action ;
    BEGIN
    p := k ; k := n ; n := p ;
    END ;
BEGIN
k := 3 ; n := 4 ;
affiche ;
action ; affiche ;
action ; affiche ;
READLN ;
END.
```

2 Variables locales dans une procédure

De même que pour les fonctions, les procédures peuvent avoir besoin de variables LOCALES, c'est-à-dire qu'on utilise dans la procédure mais plus après. Elles sont alors déclarées au cours de la procédure.

Exercice 24.3

Que fait le programme suivant ?

```
PROGRAM variables_locales ;
    VAR k,n : INTEGER ;
PROCEDURE action ;
    VAR p : INTEGER ;
    BEGIN
    p := k ; k := n ; n := p ;
    END ;
BEGIN
k := 3 ; n := 4 ;
WRITELN(k,' ',n) ;
action ; WRITELN(k,' ',n) ;
action ; WRITELN(k,' ',n) ;
READLN ;
END.
```

Remarquons qu'une variable locale n'existe que dans la procédure où elle a été déclarée. Le programme principal n'a, lui, jamais accès à une variable locale d'une procédure. Une procédure n'a jamais accès aux variables locales d'une autre procédure : le but des variables locales est d'alléger le programme et de le rendre plus clair.

Procédures à paramètres

Exercice 24.4

Que fait le programme suivant ?

```
PROGRAM faisons_des_produits ;
    VAR x,y,z,a,b,c,d : REAL ;
PROCEDURE produit ;
    BEGIN
    z := x*y ;
    END ;
BEGIN
WRITELN('Donner a et b') ;
READLN(a,b) ;
x := a ; y := b ;
produit ;
c:= z ;
x := a-1 ; y := b+1 ;
produit ;
d := z ;
WRITELN('c=', c ,' et d=', d);
READLN ;
END.
```

Remarquons que dans le programme précédent :

- les variables x et y sont à la fois dans la procédure et dans le programme principal : possibilité d'erreur.
- Il y a deux sortes de paramètres dans la procédure `produit` : les paramètres donnés (x et y) et les résultats de la procédure (z).

On va donc déclarer des paramètres à la procédure :

Exercice 24.5

Que fait le programme suivant ?

```
PROGRAM faisons_des_produits_plus_simplement ;
    VAR a,b,c,d : REAL ;
PROCEDURE produit( x,y : REAL ; VAR z : REAL) ;
    BEGIN
    z := x*y ;
    END ;
BEGIN
WRITELN('Donner a et b') ;
READLN (a,b) ;
produit(a,b,c) ;
produit(a-1,b+1,d) ;
WRITELN('c=', c ,' et d=', d);
READLN ;
END.
```

Les règles pour écrire les paramètres dans une procédure :

- Les paramètres sont donnés **dans un ordre précis** dans la **parenthèse** qui suit le nom de la procédure. Ils sont séparés par une virgule. Attention à bien déclarer le type correspondant au paramètre.
- Deux sortes de passage par paramètres :
  - ⋆ Le passage par valeur.
    Comme pour une fonction, à l'appel, le paramètre est une variable ou une expression. Cette valeur est transmise : elle sert à initialiser la variable correspondante dans la procédure.
  - ⋆ Le passage par référence.
    A l'appel , le paramètre est uniquement une variable (jamais une expression). C'est la référence du paramètre, autrement dit, c'est l'adresse mémoire qui est transmise et non sa valeur ; la variable utilisée dans la procédure est en fait la variable de l'appel mais sous un autre nom.
- Le symbole `VAR` indique si le passage se fait par valeurs (pas de `VAR`) ou par référence (avec `VAR`). Donc bien vérifier si ce mot clé est utilisé ou non.
- On ne déclare pas en variable locale un des paramètres de la procédure

Exercice 24.6

Soit la suite définie par :

$$\begin{cases} u_0 = 1 \\ \forall n \in \mathbb{N},\ u_{n+1} = 5u_n + n^2 \end{cases}$$

Ecrire deux programmes nommés `calcul1` et `calcul2` utilisant une et une seule des procédures suivantes pour calculer u_{100}.
Pourquoi ne peut-on pas utiliser les deux autres ?

```
PROCEDURE iteration_1 ;
    BEGIN
    u := 5*u+k2 ;
    END ;

PROCEDURE iteration_2(u,k2 : REAL);
    BEGIN
    u := 5*u+k2 ;
    END ;

PROCEDURE iteration_3(VAR u,k2 : REAL);
    BEGIN
    u := 5*u+k2 ;
    END ;

PROCEDURE iteration_4(k2 : REAL ; VAR u : REAL);
    BEGIN
    u := 5*u+k2 ;
    END ;
```