Procédures et fonctions

Informatique 4

LMD3 (gestion)

Par LAKEHAL K.

Rappel

- Ce cours rappelle les notions importantes de procédure et de fonction pour mieux comprendre la programmation d'événements sous Delphi.

Les procédures

- Une procédure est un sous-programme.
- Ecrire des procédures permet de découper un programme en plusieurs morceaux.
- Chaque procédure définit une nouvelle instruction, que l'on peut appeler en tout endroit du programme.
- On peut ainsi réutiliser le code d'un sous-programme.

Procédure sans paramètre (1/4)

- **Principe**

- Il s'agit simplement de donner un nom à un groupe d'instructions.

- Ensuite, l'appel de ce nom à divers endroits du programme provoque à chaque fois l'exécution de ce groupe d'instructions.

Procédure sans paramètre (2/4)

- Exemple d'une procédure:

```
PROCEDURE Echange_xy;
begin
{ Corps de la procédure }
t := x;
x := y;
y := t;
end;
```

Procédure sans paramètre (3/4)

- Exemple d'appel de procédure dans le programme principale:

```
BEGIN     { Programme principal }
x := 3; y := 4;
writeln (x, ' ', y);
Echange_xy;       { 1er appel de la procédure }
writeln (x, ' ', y);
Echange_xy;       { 2eme appel de la procédure }
writeln (x, ' ', y);
END.
```

Procédure sans paramètre (4/4)

- Ce programme affiche

```
3   4
4   3
3   4
```

Remarques :

- Le nom de la procédure est un identificateur.
- On déclare toute procédure avant le BEGIN du programme principal.

Appels de procédures (1/4)

- On peut très bien appeler une procédure P1 depuis une procédure P2,
- Mais il faut que la procédure P1 aie été déclarée avant la procédure P2.

Appels de procédures (2/4)

- Exemple donnant le même résultat.

```
PROGRAM exemple2;
VAR x, y, t : integer;
   PROCEDURE Affiche_xy;
   begin
   writeln (x, '    ', y);
   end;
```

Appels de procédures (3/4)

```
PROCEDURE Echange_xy;
begin
t := x;
x := y;
y := t;
Affiche_xy;
end;
```

Appels de procédures (4/4)

```
BEGIN
x := 3;
y := 4;
Affiche_xy;
Echange_xy;
Echange_xy;
END.
```

Remarque :

- On peut aussi appeler une procédure depuis elle-même : c'est la récursivité.

Variables locales (1/3)

- Les objets du programme qui ne sont utiles que dans la procédure peuvent être définis dans les déclarations locales de la procédure.

- Ce sont nommées des variables locales.

- Reprenons l'exemple 1 et changeons t :

Variables locales (2/3)

```
PROGRAM exemple3;
VAR x, y : integer;
   PROCEDURE Echange_xy;
   VAR t : integer;
   begin
   t := x;        x := y;          y := t;
   end;
BEGIN
{ ... }
END.
```

Declaration locale

Variables locales (3/3)

- Une variable déclarée localement n'existe que pendant l'exécution de la procédure, et ne sert qu'à cette procédure.
- Le programme principal n'a jamais accès à une variable locale de procédure.
- Une procédure n'a jamais accès à une variable locale d'une autre procédure.

Variables globales

- Les variables déclarées dans le VAR du programme principal sont appelées variables globales.
- Elles existent pendant toute la durée du programme et sont accessible de partout.
- Une variable locale à une procédure P, portant le même nom x qu'une variable globale, masque la variable globale pendant l'exécution de la procédure P.

Procédure paramétrée (1/5)

- Ecrivons une procédure Produit qui calcule z = xy.

```
PROGRAM exemple4;
VAR x, y, z, a, b, c, d : real;

    PROCEDURE Produit;
    BEGIN
    z := x * y;
    END;
```

Procédure paramétrée (2/5)

- {On veut se servir de la procédure Produit pour calculer c = a*b et d = (a-1)*(b+1)}

```
BEGIN
write (‘entrez les valeurs de a b ? '); readln (a, b);
x := a;      y := b;
Produit;
c := z;
x := a-1;    y := b+1;
Produit;
d := z;
writeln ('c = ', c, ' d = ', d);
END.
```

Données

Résultat

Données

Résultat

Procédure paramétrée (3/5)

- ***Remarques***
- L'écriture est un peu lourde.

- Deux sortes de paramètres : données et résultats.

- ***Solution***
- La solution élégante consiste à déclarer des paramètres à la procédure :

Procédure paramétrée (4/5)

```
PROGRAM exemple5;
VAR a, b, c, d : real;
    PROCEDURE Produit (x, y : real; var z : real);
    BEGIN
    z := x * y;
    END;
BEGIN
write (‘entrez les valeurs de a et b ');
readln (a, b);
Produit (a, b, c);
Produit (a-1, b+1, d);
writeln ('c = ', c, ' d = ', d);
END.
```

Paramètres

Passage de paramètres

Procédure paramétrée (5/5)

- À l'appel, on donne des paramètres dans les parenthèses, séparées par des virgules, et dans un certain ordre (ici a puis b puis c).

1. L'exécution de la procédure commence ;
2. la procédure reçoit les paramètres et identifie chaque paramètre à une variable dans le même ordre (ici x puis y puis z).

- Les types doivent correspondre ; ceci est vérifié à la compilation.

Les fonctions

- Une fonction est une procédure "qui ramène un résultat" d'un type simple qu'on précise dans la déclaration de la fonction.

- Pour "ramener" le résultat, il suffit d'ajouter une ligne où on affecte le résultat au nom de la fonction.

Les fonction: syntaxe

Program *nom_de_programme* ;

function *nom_de_fonction*(noms de variables : types) : type;
Begin
...
nom_de_fonction := résultat
...
End ;

BEGIN
variable := nom_de_fonction (noms d'autres variables ou leurs valeurs) ;
END.