

Les procédures paramétrées

Ch. Lasou, N.E. Oussous et E. Wegrzynowski

Licence ST-A, USTL - API1

22 janvier 2007

- ▶ Procédure = action sur l'environnement : affichage de valeurs de paramètres, lecture de valeurs de paramètres, modification de valeurs de paramètres ;
- ▶ ces valeurs de paramètres peuvent être fournies aux procédures : ce sont des données passées en entrée à la procédure ;
- ▶ elles peuvent être produites par la procédure : ce sont des résultats récupérés en sortie de la procédure ;
- ▶ elles peuvent aussi être fournies à et modifiées par la procédure : ce sont à la fois des données et des résultats de la procédure.

Il existe donc trois modes de passages de paramètres : paramètre en entrée, paramètre en sortie, paramètre en entrée/sortie.

Introduction

Paramètre en entrée

Paramètre en sortie

Paramètre en entrée/sortie

Procédures ou fonctions

Exemples

Paramètre en entrée affichage d'une valeur

```
{ n = 7 }  
afficher ( 2 * n + 1 );  
{ affichage de 15 }
```

Paramètre en sortie lecture d'une valeur

```
{ n = ??? }  
lire ( n );  
{ n = la valeur lue }
```

Paramètre en entrée/sortie modification d'une valeur

```
{ n = 6 }  
incrémenter ( n );  
{ n = 7 }
```

Objectif du chapitre

Présenter

1. l'écriture en PASCAL de procédures selon les différents modes de passage de paramètres
2. et l'utilisation de ces procédures.

Remarque (1/2)

Lorsqu'un paramètre formel est déclaré en entrée à l'aide du mot-clé **const**, il n'est pas possible de le modifier dans le corps de la procédure.

Le compilateur FREE PASCAL refuse la déclaration suivante

```
procedure incorrecte(const n : INTEGER);
begin
  n := 1;
end { incorrecte };
```

en indiquant le message

Error : Can't assign values to const variable

Déclaration de paramètre formel en entrée

Syntaxe en PASCAL

Utilisation du mot-clé **const** placé devant le paramètre formel

Exemple

```
// affichage d'une valeur de type jour
procedure afficherJour(const j : JOUR);
```

Remarque (2/2)

Pour les paramètres en entrée, il est possible de les déclarer sans mettre le mot-clé **const**.

Dans ce cas, la modification est autorisée dans le corps de la procédure, mais les éventuelles modifications ne persistent pas à la fin de son exécution

```
procedure correcte(n : INTEGER);
begin
  n := n+1 ;
  writeln( 'n=␣', n ) ;
end { correcte };

...

{n = 2}
correcte(n); // affiche : n = 3
{n = 2}
```

Paramètre effectif en entrée

À l'appel d'une procédure ayant un paramètre en entrée, le paramètre effectif correspondant peut être toute expression d'un type compatible.

```
var j : JOUR;

...

afficherJour(LUNDI);

j := MARDI;
afficherJour(j);
afficherJour(succ(j));
```

Remarque

La valeur d'un paramètre formel déclaré en sortie ne doit pas être utilisée dans le corps de la procédure avant d'avoir été initialisé.

```
procedure incorrecte(out n : INTEGER);
begin
  n := 3*n+1;
end { incorrecte };

procedure correcte(out n : INTEGER);
begin
  n := 1;
  n := 3*n+1;
end { correcte };
```

Néanmoins le compilateur FREE PASCAL ne signale rien pour la procédure incorrecte

Déclaration de paramètre formel en sortie

Syntaxe en PASCAL

Utilisation du mot-clé **out** placé devant le paramètre formel

Exemple

```
// lecture d'une valeur de type jour
procedure lireJour(out j : JOUR);
```

Paramètre effectif en sortie

- ▶ À l'appel d'une procédure ayant un paramètre en sortie, le paramètre effectif correspondant doit être une variable d'un type compatible.

```
var j : JOUR;

lireJour(j);
```

- ▶ Si le paramètre effectif est une expression, le compilateur refuse

```
lireJour(LUNDI);
```

avec le message d'erreur

Error : Variable identifier expected

Déclaration de paramètre formel en entrée/sortie

Syntaxe en PASCAL

Utilisation du mot-clé **var** placé devant le paramètre formel

Exemple

```
// modifie la valeur de j en succ(j)
procedure incrementer(var j : JOUR);
```

Passage des paramètres de fonction

- ▶ Les trois modes de passage existent aussi pour les fonctions en PASCAL (ce qui n'est pas le cas dans tous les langages);
- ▶ mais on n'utilisera que le mode de passage en entrée (avec ou sans **const**).

Paramètre effectif en entrée/sortie

- ▶ À l'appel d'une procédure ayant un paramètre en entrée/sortie, le paramètre effectif correspondant doit être une variable d'un type compatible.

```
var j : JOUR;
```

```
{j = LUNDI}
incrémenter(j);
{j = MARDI}
```

- ▶ Si le paramètre effectif est une expression, le compilateur refuse

```
incrémenter(LUNDI);
```

avec le message d'erreur

Error : Variable identifier expected

Procédures au lieu de fonctions

- ▶ Toute fonction peut être réalisée par une procédure.

```
function f(x : INTEGER) : STRING;
...
```

peut être écrit sous la forme

```
procedure f(const x : INTEGER; out y : STRING);
...
```

- ▶ Toutefois, les fonctions présentent une meilleure souplesse dans l'écriture des programmes.
- ▶ Mais le calcul effectué par une procédure peut produire plusieurs résultats ce qu'une fonction ne peut pas faire

Calcul du quotient et reste

Division Euclidienne

On se donne deux entiers a et $b \neq 0$. On veut calculer le quotient et le reste de la division euclidienne de a par b .

```
procedure quorem(const a,b : INTEGER; out q,r : INTEGER);  
begin  
  r := a ;  
  q := 0 ;  
  while r > b do begin  
    r := r - b ;  
    q := q + 1 ;  
  end; { while }  
end { quorem } ;
```