Cours 6

M.Adam – JF.Kamp – S.Letellier – F.Pouit

1er août 2016

Table des matières

1 Les procédures

1.1 Exemple introductif

Soit le code suivant :

```
##
# Saisie des notes en informatique et calcul de la moyenne
# @author M.Adam
##
algo MoyenneInfo
principal
var
    reel m1101, m1102, m1103, moyenne;
debut
    repeter
        saisir("M1101 : Introduction aux systèmes informatiques", @m1101);
    jusqua (0<=m1101 et m1101<=20)
    finrepeter

    repeter
        saisir("M1102 : Introduction à l'algorithmique et à la programmation", @m1102);
    jusqua (0<=m1102 et m1102<=20)
    finrepeter

    repeter
        saisir("M1103 : Structures de données et algorithmes fondamentaux", @m1103);
    jusqua (0<=m1103 et m1103<=20)
    finrepeter

    moyenne := (3.5*m1101 + 3.5*m1102 + 2.5*m1103)/9.5;
    afficher("Moyenne = "+moyenne);

fin
```

Remarques

- le même code est écrit plusieurs fois,
- même avec une version simplifiée le code est longue à lire,
- le risque d'erreurs est multiplié.

1.2 Définition d'une procédure

Une procédure est une suite d'instructions réalisant une action et pouvant être utilisée plusieurs fois dans un programme.

Une manière de factoriser le code est d'utiliser les procédures.

Une procédure doit être **déclarée** avant d'être **utilisée** ou **appelée**.

Procédures prédéfinies
Nous avons déjà utilisé des procédures prédéfinies de TestAlgo :

- `alaligne()`
- `afficher()` et `afficherln()`
- `saisir()`

1.3 Les paramètres

Un paramètre est une valeur ou une variable passée en argument d'une procédure à son appel et déclarée à la définition de la procédure.

- `alaligne()` n'a aucun paramètre,
- `afficher()` et `afficherln()` ont un paramètre,
- `saisir()` a deux paramètres

1.3.1 Paramètres en entrée ou en sortie

Un paramètre est dans l'une des trois catégories :

- uniquement en entrée,
- uniquement en sortie,
- en entrée et en sortie.

1.3.2 Passage par valeur

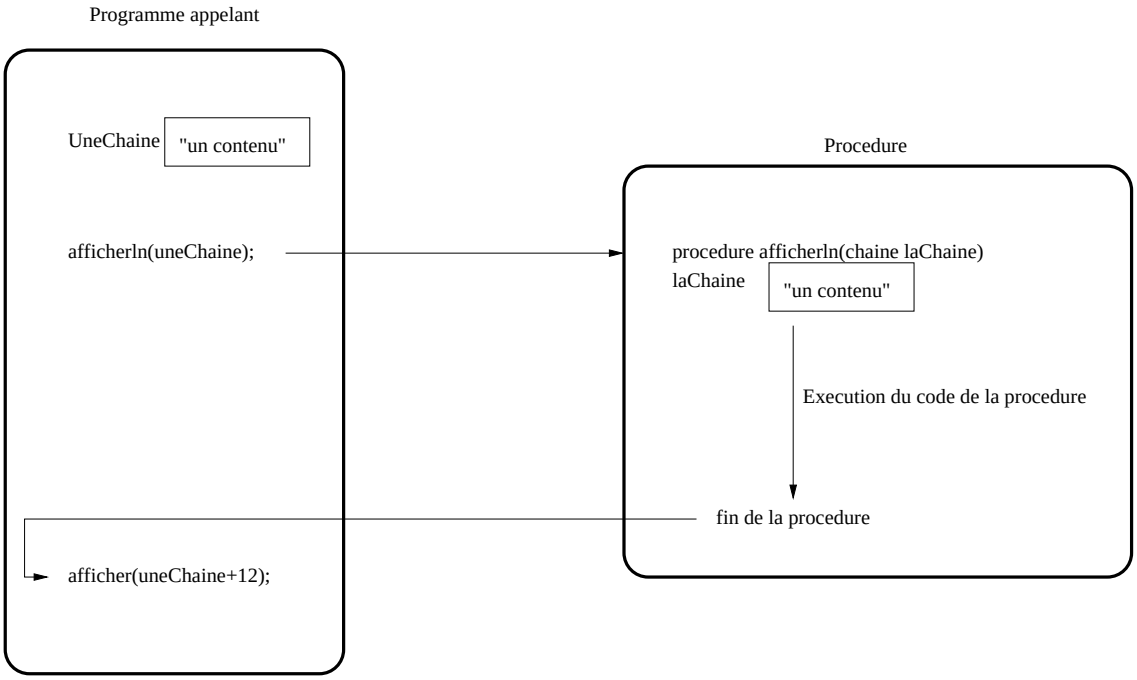
C'est une valeur qui est passée en paramètre :

```
##
# Exemple de passage de paramètres par valeur
# @author M.Adam
##
algo Valeur
principal
var
    chaine uneChaine;
debut
    afficherln("une valeur");
    afficherln(123);
    uneChaine := "un contenu";
    afficherln(uneChaine);
    afficherln(uneChaine+12);
fin


TestAlgo - Interprétation engagée.
une valeur
```

```
123
un contenu
un contenu12

TestAlgo - Fin de l'interprétation.
```



1.3.3 Passage par référence

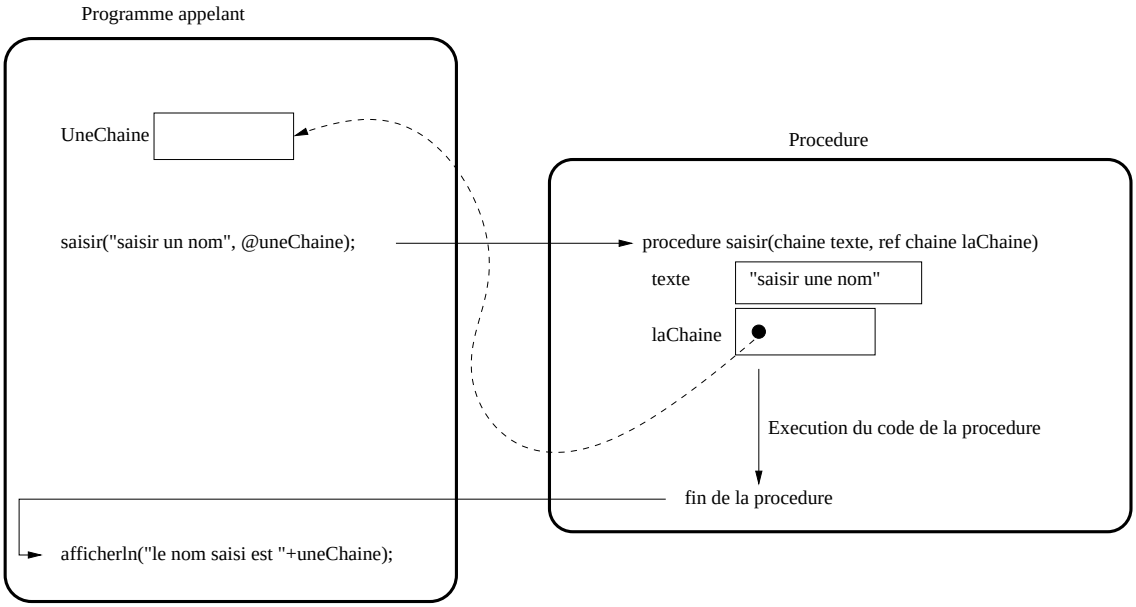
C'est une "adresse", une rérérence, de la variable que est passée en paramètre :

```
##
# Exemple de passage de paramètres par référence
# @author M.Adam
##
algo Reference
principal
var
    chaine uneChaine;
debut
    saisir("saisir un nom",@uneChaine);
    afficherln("Le nom saisi est "+uneChaine);
fin


TestAlgo - Interprétation engagée.
Adam
```

```
Le nom saisi est Adam

TestAlgo - Fin de l'interprétation.
```



1.3.4 Par valeur ou par référence ?

Utilisation	Type de paramètre
paramètre en entrée uniquement	par valeur
paramètre en sortie uniquement	par référence
paramètre en entrée et en sortie	par référence

Attention : en TestAlgo, un tableau est toujours passé par référence.

1.3.5 Petite question ?

Et que se passerait-il si l'on passait un paramètre d'entrée par une référence ?

1.4 Reprise de l'exemple

La bonne manière d'écrire le programme de calcul de la moyenne Info serait :

```
##
# Saisie des notes en informatique et calcul de la moyenne
# @author M.Adam
##
algo MoyenneInfo
```

```
principal
var
    reel m1101, m1102, m1103, moyenne;
debut
    saisirNote("M1101 : Introduction aux systèmes informatiques", @m1101);
    saisirNote("M1102 : Introduction à l'algorithmique et à la programmation", @m1102);
    saisirNote("M1103 : Structures de données et algorithmes fondamentaux", @m1103);

    moyenne := (3.5*m1101 + 3.5*m1102 + 2.5*m1103)/9.5;
    afficher("Moyenne = "+moyenne);

fin
```

- le code est plus lisible
- il est possible de tester de **manière unitaire** `saisirNote()`

1.5 Déclaration

La déclaration d'une procédure consiste à définir ses instructions, son code, ainsi que ses paramètres.

```
procedure maProc ([par_ref] type param1, [par_ref] type param2, ... )
var
    type var1;
    type var2;
    ...
debut
    instructions;
    ...
    ...
fin
```

Remarques

- Le mot clé `par_ref` indique le paramètre est passé par référence, donc en sortie ou en entrée-sortie.
- Les variables définies dans une procédure ne sont visibles et utilisables que dans la procédure (portée).

1.6 Retour sur l'exemple

```
##
# Saisie des notes en informatique et calcul de la moyenne
# @author M.Adam
##
algo MoyenneInfo
principal
var
    reel m1101, m1102, m1103, moyenne;
debut
    saisirNote("M1101 : Introduction aux systèmes informatiques", @m1101);
```

```
    saisirNote("M1102 : Introduction à l'algorithmique et à la programmation", @m1102);
    saisirNote("M1103 : Structures de données et algorithmes fondamentaux", @m1103);

    moyenne := (3.5*m1101 + 3.5*m1102 + 2.5*m1103)/9.5;
    afficher("Moyenne = "+moyenne);
fin

##
# saisir une note et vérifier sa valeur entre 0 et 20
# @param titre intitulé de la note
# @param note  note saisie durant la procédure
##
procedure saisirNote (chaine titre, par_ref reel note)
debut
    repeter
        saisir(titre, @note);
    jusqua (0<=note et note<=20)
    finrepeter
fin
```

2 Les fonctions

2.1 Exemple introductif

Soit le code suivant :

```
##
# affichage de 3^5 + 4^7
# @author M.Adam
##
algo Exposant
principal
var
    reel res1, res2;
debut
    exposant(3, 5, @res1);
    exposant(4, 7, @res2);
    afficher("Le résultat de 3^5 + 4^7 est ");
    afficherln(res1+res2);
fin
##
# Calcul de x^n
# @param x valeur à monter à l'exposant
# @param n valeur de l'exposant
# @param res résultat de x^n
##
procedure exposant (reel x, entier n, par_ref reel res)
var
    entier i;
```

```
debut
    res := 1;
    pour i:=1 à n pas 1
        res := res*x;
    finpour
fin
```

Remarques

- La procédure `exposant` ne rend qu'un résultat.
- Les autres paramètres de `exposant` sont passés par valeur
- Il faut récupérer la résultat de `exposant` pour enchaîner ensuite l'addition.

2.2 Définition d'une fonction

Une fonction est une "procédure" qui rend un seul résultat et dont les autres paramètres sont passés par valeur ou par référence.

Quelques fonctions prédéfinies

- `entier modulo (entier dividende, entierdiviseur);`
- `entier arrondi (reel valeur);`
- `entier longueur (chaine ch);`
- `reel aleatoire();`
- `reel racine (reel valeur).`

Et quelques autres encore...

2.3 Déclaration d'une fonction

```
fonction maFonc ([par_ref] type param1, [par_ref] type param2, ...): type
var
    type var1;
    type ret;
    ...
debut
    instructions;
    ...
    retourne ret;
fin
```

- Les paramètres sont passés par valeur ou par référence,
- les variables déclarées dans la fonction ne sont visibles et utilisables que dans la fonction (portée),
- la fonction se termine obligatoirement par `retourne ret;`
- le type de `ret` est le même que celui retourné par la fonction et défini à la première ligne de la fonction, derrière les deux points (:).

2.4 Retour à l'exemple

```
##
# affichage de 3^5 + 4^7
# @author M.Adam
##
algo Exposant
principal
debut
    afficher("Le résultat de 3^5 + 4^7 est ");
    afficherln(exposant(3, 5)+exposant(4, 7));
fin

##
# Calcul de x^n
# @param x valeur à monter à l'exposant
# @param n valeur de l'exposant
# @return la valeur de x exposant n
##
fonction exposant (reel x, entier n): reel
var
    entier i;
    reel   res;
debut
    res := 1;
    pour i:=1 à n pas 1
        res := res*x;
    finpour
    retourne res;
fin
```

2.4.1 Ou encore

```
algo Exposant
principal
var
  reel res1, res2;
debut
  res1 := exposant(3, 5);
  res2 := exposant(4, 7):
  afficher("Le résultat de 3^5 + 4^7 est ");
  afficherln(res1+res2);
fin
```

2.4.2 Petit exercice

Et comment réécrire la procédure `saisirNote` pour qu'elle devienne une fonction ?

2.5 Procédure ou fonction ?

- Les procédures suffisent.
- Écrire une fonction c'est simplifier l'écriture du code de l'appelant ainsi que sa lisibilité.

2.6 Utilisation des procédures et des fonctions

- Quand une séquence d'instructions se répète
- Pour décomposer un gros problème en sous problèmes plus simples
- Pour enrichir le langage avec de nouveaux "types" ou "objets" (voir java)
- Pour pouvoir tester un programme plus simplement
- Pour rendre le code plus simple à lire et à modifier

3 Notions complémentaires

3.1 Signature

La signature d'une fonction, d'une procédure, en TestAlgo est définie de manière unique par :

- son nom
- la déclararation de ses paramètres

Par exemple :

```
fonction exposant (reel x, entier n): reel
```

La signature suffit au programmeur pour appeler la fonction ou la procédure, ainsi que pour récupérér les résultats.

Mais pour que le programmeur puisse utiliser une fonction ou une procédure il lui faut également son rôle :

```
##
# Calcul de x^n
# @param x valeur à monter à l'exposant
# @param n valeur de l'exposant
# @return la valeur de x exposant n
##
fonction exposant (reel x, entier n): reel
```

3.2 Portée

Les variables déclarées dans la signature d'une fonction, d'une procédure ou dans le bloc `var`, ne sont utilisables que dans la fonction ou la procédure.

```
##
# Calcul de x^n
# @param x valeur à monter à l'exposant
# @param n valeur de l'exposant
# @return la valeur de x exposant n
##
fonction exposant (reel x, entier n): reel
var
    entier i;
    reel   res;
debut
    res := 1;
    pour i:=1 à n pas 1
        res := res*x;
    finpour
    retourne res;
fin
```

Les variables `x`, `n`, `i`, `res` ne sont utilisables que dans la fonction `exposant`.

Cette notion est appelée la **portée** des variables.

De même, les variables déclarées dans le `var` du programme principal ne sont visibles que dans celui-ci.

4 Les tests

4.1 Tests Fonctionnels ou unitaires

- **fonctionnels** : se mettre à la place de l'utilisateur du programme et tester que toutes les fonctionnalités sont correctes
- **unitaires** : tester une partie du code, des fonctions ou des procédures pour vérifier qu'elles donnent les bons résultats.

5 Et pour en finir

5.1 A retenir

- Notion de procédures et de fonctions
- Passage de paramètres par valeur ou par référence
- Tests fonctionnels et unitaires

BONNE CONTINUATION !