ASD I

LFSI1/LAI1

A-U: 2017/2018

4

LES SOUS-PROGRAMMES (PROCÉDURE ET FONCTION)

Introduction

- En cas de traitements similaires
  - Il faut séparer les traitements similaires du corps du programme
- En cas de problèmes complexes
  - Il faut subdiviser le problème en une série de plus petits problèmes
- Conséquences :
  - Lisibilité assurée. On dit que le programme devient modulaire.
  - Pour la maintenance, il suffit de faire une seule modification dans le groupe d'instructions pour que cette modification prenne effet dans la totalité de l'application.
- Le corps du programme s'appelle alors le programme principal.
- Les groupes d'instructions auxquels on a recours s'appellent des fonctions et des procédures. Ils forment ce qu'on appelle des sous-programmes.

Les fonctions

- Format général d'une fonction :

```
Fonction Nom_Fonction(Arg1 : Type, Arg2 : Type, … ) : Type

Var : [Variables Locales]
Début
        …
        [Traitements]
        …
        Renvoyer [Valeur Renvoyee]
Fin
```

Les fonctions

- Exemple: une fonction qui renvoie la moyenne de deux entier.

```
Fonction Moyenne(A : entier, B : entier) : réel
Var :
        Moy : réel
Début
        Moy ← (A + B) / 2
        Renvoyer Moy
Fin
```

```
Fonction Moyenne (A : entier, B : entier) : réel
Début
        Moyenne ← (A + B) / 2
Fin
```

Le nom de la fonction joue un double rôle, c'est à la fois l'identifiant de la fonction et une variable locale.

Les procédures

- Une fonction doit renvoyer une et une seule valeur
- Comment faire dans le cas où on veut ne pas renvoyer des valeurs ou renvoyer plusieurs valeurs!!
  - Solution : Utiliser les procédures
- Format général d'une procédure :

```
Procédure Nom_Procédure (Arg1 : Type, Arg2 : Type, … )
Var
        [Variables Locales]
Début
        [Traitements]

Fin
```

Les procédures

- De même qu'avec les fonctions, les valeurs qui circulent depuis l'algorithme appelant vers la procédure appelée se nomment des **arguments**, ou des **paramètres en entrée** de la procédure.
- Dans une procédure, on peut être amené a vouloir renvoyer des résultats vers l'algorithme principal. Or, la procédure, en tant que telle, **ne renvoie rien du tout** !!
- Les résultats que la procédure doit transmettre devront être véhiculés par des paramètres. Il s'agit de **paramètres en sortie**.

Les procédures

- Exemple : Procédure sans paramètre

```
Procédure Etoile ( )
VAR
        i : entier
DÉBUT
    Pour i de 1 à 10 faire
        Écrire ( " * " )
    FinPour
FIN
```

Les procédures

- Exemple : Procédure avec paramètre

```
Procédure Etoile ( nb: entier)
VAR
        i : entier
DÉBUT
    Pour i de 1 à nb faire
        Écrire ( " * " )
    FinPour
FIN
```

Les procédures

- Exemple complet

```
ALGORITHME PROC
CONST
        titre = "Algorithmique" : chaîne de caractères
        nb_et =15: entier
Procédure Etoile ( nb:entier)
VAR
        i : entier
DÉBUT
    Pour i de 1 à nb faire
        Écrire ( " * " )
    FinPour
FIN
DÉBUT  //Programme principal
        Ecrire (titre)
        Etoile(nb_et ) /* Appel de la procédure Etoile*/
FIN
```

Passage des paramètres

- Types de paramètres:
  - Paramètres fictifs : figurent dans l'entête de la déclaration de la procédure ou de la fonction
  - Paramètres réels : figurent dans l'instruction d'appel du sous-programme et sont substitués aux paramètres fictifs au moment de l'appel.
- Remarques:
  - Les paramètres réels et les paramètres fictifs doivent s'accorder du point de vue nombre et ordre.
  - Leurs types doivent être identiques

Passage des paramètres

- Types de paramètres:

```
ALGORITHME PROC
CONST
        titre = "Algorithmique" : chaîne de caractères
        nb_et =15: entier
PROCÉDURE Etoile (nb:entier)
    VAR
        i : entier
    DÉBUT
        Pour i de 1 à nb faire
                Écrire ( " * " )
        FinPour
    FIN
DÉBUT  //Programme principal
        Ecrire (titre)
        Etoile(nb_et) /* Appel de la procédure Etoile*/
FIN
```

Paramètre fictif

Paramètre réel

Passage des paramètres

- Types de de passages de paramètres:
  - Par Valeur : Le passage de paramètres par valeur permet au programme appelant de transmettre **une valeur** à la procédure (ou fonction) appelée.
  - Par référence (ou adresse ou variable) : ce type de passage permet au programme appelant de transmettre une valeur à **la procédure** appelée et **vice-versa**.
    - **Ce type est utilisé pour renvoyer des valeurs par une procédure**
- Dans une fonction, le passage de tous les paramètres se fait par valeur

Passage des paramètres par valeur

- Le transfert d'information est effectué dans un seul sens, du programme principal vers la procédure.
- Toute modification du paramètre fictif est sans conséquent sur le paramètre réel.
- Exemple

```
ALGORITHME PASSAGE_VALEUR
VAR
x : entier
Procédure ESSAI ( a : entier )
DÉBUT
        a ← 3 * a
        Écrire ( "Le paramètre a =" , a )
FIN
```

```
DÉBUT // Programme principal
        Ecrire ( "Donner la valeur de x : " )
        Lire (x)
        Ecrire ( "Avant appel x = " , x )
        ESSAI ( x ) /* Appel de la procédure*/
        Ecrire ( "Après appel x = " , x )
FIN
```

Passage des paramètres par valeur

- Exemple

```
ALGORITHME PASSAGE_VALEUR
VAR
x : entier
Procédure ESSAI ( a : entier )
DÉBUT
        a ← 3 * a
        Écrire ( "Le paramètre a =" , a )
FIN
```

```
DÉBUT // Programme principal
        Ecrire ( "Donner la valeur de x : " )
        Lire (x)
        Ecrire ( "Avant appel x = " , x )
        ESSAI ( x ) /* Appel de la procédure */
        Ecrire ( "Après appel x = " , x )
FIN
```

X= 5	Prog. Principal	Procédure Essai
Appel de « ESSAI »		
Exécution de « ESSAI»		
Après « ESSAI »		

Passage des paramètres par valeur

- Exemple

```
ALGORITHME PASSAGE_VALEUR
VAR
x : entier
Procédure ESSAI ( a : entier )
DÉBUT
        a ← 3 * a
        Écrire ( "Le paramètre a =" , a )
FIN
```

```
DÉBUT // Programme principal
        Ecrire ( "Donner la valeur de x : " )
        Lire (x)
        Ecrire ( "Avant appel x = " , x )
        ESSAI ( x ) /* Appel de la procédure */
        Ecrire ( "Après appel x = " , x )
FIN
```

X= 5	Prog. Principal	Procédure Essai
Appel de « ESSAI »	x = 5	a = 5
Exécution de « ESSAI»	x = 5	a = 15
Après « ESSAI »	x = 5	a = 15

Passage des paramètres par variable

- Le transfert d'information est effectué dans les deux sens, du programme principal vers la procédure et vice-versa.
- Toute modification du paramètre fictif entraîne automatiquement la modification de la valeur du paramètre réel.
- Exemple

```
ALGORITHME PASSAGE_VALEUR
VAR
x : entier
Procédure ESSAI ( VAR a : entier )
DÉBUT
        a ← 3 * a
        Écrire ( "Le paramètre a =" , a )
FIN
```

```
DÉBUT // Programme principal
        Ecrire ( "Donner la valeur de x : " )
        Lire (x)
        Ecrire ( "Avant appel x = " , x )
        ESSAI ( x ) /* Appel de la procédure*/
        Ecrire ( "Après appel x = " , x )
FIN
```

Passage des paramètres par variable

- Exemple

```
ALGORITHME PASSAGE_VALEUR
VAR
x : entier
Procédure ESSAI ( VAR a : entier )
DÉBUT
        a ← 3 * a
        Écrire ( "Le paramètre a =" , a )
FIN
```

```
DÉBUT // Programme principal
        Ecrire ( "Donner la valeur de x : " )
        Lire (x)
        Ecrire ( "Avant appel x = " , x )
        ESSAI ( x ) /* Appel de la procédure */
        Ecrire ( "Après appel x = " , x )
FIN
```

X= 5	Prog. Principal	Procédure Essai
Appel de « ESSAI »		
Exécution de « ESSAI»		
Après « ESSAI »		

Passage des paramètres par variable

- Exemple

```
ALGORITHME PASSAGE_VALEUR
VAR
x : entier
Procédure ESSAI ( VAR a : entier )
DÉBUT
        a ← 3 * a
        Écrire ( "Le paramètre a =" , a )
FIN
```

```
DÉBUT // Programme principal
        Ecrire ( "Donner la valeur de x : " )
        Lire (x)
        Ecrire ( "Avant appel x = " , x )
        ESSAI ( x ) /* Appel de la procédure */
        Ecrire ( "Après appel x = " , x )
FIN
```

X= 5	Prog. Principal	Procédure Essai
Appel de « ESSAI »	x = 5	a = 5
Exécution de « ESSAI»	x = 15	a = 15
Après « ESSAI »	x = 15	a = 15