

Université Abdelmalek Essaâdi
Ecole nationale des sciences appliquées d'Al Hoceima
(ENSAH)

Algorithmique et programmation

Préparé et présenté par
Mr. Ouazzani Chahdi

Année universitaire: 2018/2019

1

Les fonctions et les procédures

2

1-Présentation

❖ Ce chapitre est présenté en se basant sur deux exemples :

Exemple 1

Ecrire un algorithme Factoriel2Nombres qui permet de calculer et afficher le factoriel de deux nombres entier saisis au clavier.

Exemple 2

Ecrire un algorithme Signe2Nombres qui permet d'afficher le signe de deux valeurs réelles saisis au clavier.

❖ Dans les deux exemples, les algorithmes doivent être écrits d'une manière optimale, c'est-à-dire optimiser le nombre d'instructions entrant en jeux.

3

❖ Discussion du premier exemple : On se contente de créer une première version :

```

Algorithme Factoriel2Nombres
Variable n, m, fact1, fact2, i : Entier
Début
  Ecrire("Donnez les valeurs de n et m :")
  Lire(n, m)
  fact1 ← 1
  Pour i allant de 2 Jusqu'à n Faire
    fact1 ← fact1*i
  FinPour
  fact2 ← 1
  Pour i allant de 2 Jusqu'à m Faire
    fact2 ← fact2*i
  FinPour
  Ecrire("n! = ", fact1, "m! = ", fact2)
Fin

```

4

❖ Dans l'algorithme précédent, on remarque que le code qui permet de calculer le factoriel d'un nombre se répète deux fois, une fois pour le nombre **n** et une deuxième fois pour le nombre **m**.

❖ Si on a un troisième nombre, alors ce code sera répété pour la troisième fois.

❖ L'idée est de trouver une façon d'écrire ce code une seule fois pour toute et de l'utiliser pour chaque nombre donné.

❖ Un tel code doit être paramétré pour qu'il s'adapte à des valeurs entières quelconques.

❖ Ce code doit retourner un résultat, c'est le résultat du calcul du factoriel.

❖ Le résultat retourné dépend bien sûr du paramètre passé au code en question.

❖ Un tel code est appelé **Fonction**.

5

❖ Discussion de l'exemple 2 : On se contente de créer une première version :

```

Algorithme Signe2Nombres
Variable x, y: Réel
Début
  Ecrire("Donnez les valeurs de x et y :")
  Lire(x, y)
  Si x >= 0 Alors
    Ecrire(x, "est positif")
  Sinon
    Ecrire(x, "est négatif")
  FinSi
  Si y >= 0 Alors
    Ecrire(y, "est positif")
  Sinon
    Ecrire(y, "est négatif")
  FinSi
Fin

```

6

- ❖ Mêmes remarques que l'exemple 1, on trouve que le code qui détermine le signe se répète deux fois.
- ❖ Alors, l'idée est de trouver une façon d'écrire ce code une seule fois pour toute et de l'utiliser pour chaque valeur donnée.
- ❖ Un tel code doit être paramétré pour qu'il s'adapte à des valeurs quelconques.
- ❖ Ce code doit ne pas retourner un résultat, il se contente juste de réaliser un traitement.
- ❖ Un tel code est appelé **Procédure**.

7

- ❖ La manipulation des fonctions et des procédures se passe par deux étapes :
 1. Déclaration et définition de la fonction ou de la procédure;
 2. L'utilisation ou l'appel.
- ❖ La déclaration et la définition consiste à décrire le comportement d'une fonction ou d'une procédure et les objets manipulés.
- ❖ La déclaration et la définition se réalise en dehors de l'algorithme(programme) principale.
- ❖ L'utilisation ou l'appel d'une fonction ou d'une procédure consiste à utiliser leurs codes dans l'algorithme principale ou dans d'autres fonctions et procédures en lui passant éventuellement des paramètres.
- ❖ Une fonction ou une procédure peuvent être appelés plusieurs fois en changeant leurs paramètres.

8

- ❖ Dans un premier temps, on peut adopter la structure suivante :


```
[ Déclaration et définition des fonctions et des procédures ]
Algorithme NomAlgorithme
[ Bloc de déclaration ]
Début
[ Traitement ]
[ Appel d'une fonction ou d'une procédure ]
[ Traitement ]
[ Appel d'une fonction ou d'une procédure ]
.
.
Fin
```

9

2-Les fonctions

- ❖ Une fonction est un sous-algorithme(sous-programme) constitué d'une suite d'instructions indépendantes.
- ❖ Elle prend éventuellement quelques paramètres et retourne un résultat.
- ❖ Une fonction se caractérise par son **nom** et le **type** de la **valeur retournée**.
- ❖ Elle peut contenir un bloc de déclaration des constantes, des variables, des tableaux, etc.



10

2.1-Déclaration et définition d'une fonction

Syntaxe :

```
Fonction NomFonction([paramètre1 : Type1, ... ,
paramètreN : TypeN]) : TypeDeRetour
[Bloc de déclaration]
Début
.
.
.
Retourner ValeurDeRetour
FinFonction
```

11

□ Avec :

- **NomFonction** : représente le nom de la fonction
- **paramètre1 : Type1, ... , paramètreN : TypeN** : les éventuelles paramètres de la fonction, pour chaque paramètre on précise le **nom** et le **type**.
- **TypeDeRetour** : représente le type de la valeur retournée par la fonction.
- **ValeurDeRetour** : représente n'importe quelle expression qui retourne une valeur du même type que **TypeDeRetour**.

2.2-Appel d'une fonction

- ❖ L'utilisation ou l'appel d'une fonction se réalise à l'aide de la syntaxe : **NomFonction** (**[paramètre1, ..., paramètreN]**)
- ❖ Puisqu'une fonction retourne une valeur alors elle peut être utilisée dans une expression ou l'on peut afficher directement son résultat.

12

❖ Deuxième version de l'algorithme **Factoriel2Nombres**

```

Fonction Factoriel(n : Entier) : Entier
Variable fact, i : Entier
Début
  fact ← 1
  Pour i allant de 2 Jusqu'à n Faire
    fact ← fact*i
  FinPour
  Retourner fact
FinFonction
Algorithme Factoriel2Nombres
Variable n, m : Entier
Début
  Ecrire("Donnez n et m :")
  Lire(n,m)
  Ecrire("n! = ",Factoriel(n)," m! = ",Factoriel(m))
Fin

```

Déclaration

Appel

13

2.3-La valeur du retour d'une fonction

- ❖ La valeur du retour d'une fonction doit être du même type que le type mentionné dans sa déclaration.
- ❖ Elle peut être une expression quelconque de même type que le type de la fonction.
- ❖ Dans le cas où la fonction retourne un tableau, ce tableau doit être géré d'une manière dynamique, c'est-à-dire qu'une fonction ne peut pas retourner un tableau statique.

Exemple

Ecrire une fonction **CréerTableau** qui permet de créer et retourner un tableau de N élément de type réel initialisé par des valeurs de 1 jusqu'à N.

Ecrire l'algorithme principale qui fait appel à cette fonction.

14

```

Fonction CréerTableau(N : Entier) : ^Réel
Variable T : ^Réel
  i : Entier

```

```

Début
  Allouer(T, N)
  Pour i allant de 0 Jusqu'à N-1 Faire
    T[i] ← i+1
  FinPour
  Retourner T

```

```

FinFonction
Algorithme Test
Variable T : ^Réel
Début
  T ← CréerTableau(10)
  Pour i allant de 0 Jusqu'à 9
    Ecrire(T[i])
  FinPour
Fin

```

15

3-Les procédures

- ❖ Une procédure est un sous-algorithme(sous-programme) constitué d'une suite d'instructions indépendantes.
- ❖ Elle prend éventuellement quelques paramètres et ne retourne aucun résultat.
- ❖ Elle se caractérise seulement par son **nom**.
- ❖ Elle peut contenir un bloc de déclaration des constantes, des variables, des tableaux, etc.



16

3.1-Déclaration et définition d'une procédure**Syntaxe :**

```

Procédure NomProcédure ([paramètre1 : Type1, ...
, paramètreN : TypeN])

```

```

[Bloc de déclaration]

```

Début

```

.
.
.

```

FinProcédure

Aucune valeur n'est retournée par cette procédure

17

❑ Avec :

- **NomProcédure** : représente le nom de la procédure
- **paramètre1 : Type1, ... , paramètreN : TypeN** : les éventuelles paramètres de la procédure, pour chaque paramètre, on précise le nom et le type.

3.2-Appel d'une procédure

- ❖ L'utilisation ou l'appel d'une procédure se réalise à l'aide de la syntaxe :

```

NomProcédure ([paramètre1, ...,paramètreN])

```

18

❖ Deuxième version de l'algorithme **Signe2Nombres**

```

Procédure Signe(x : Réel)
Début
  Si x >= 0 Alors
    Ecrire(x, "est positif")
  Sinon
    Ecrire(x, "est négatif")
  FinSi
FinProcédure
Algorithme Signe2Nombres
Variable x, y : Réel
Début
  Ecrire("Donnez x et y :")
  Lire(x,y)
  Signe(x)
  Signe(y)
Fin

```

19

Remarque 1

- Lors d'un appel d'une fonction ou d'une procédure, le nombre, l'ordre et les types des paramètres doivent nécessairement correspondre aux indications de leurs déclarations.

Remarque 2

- Les paramètres d'une fonction ou d'une procédure sont facultatifs, c'est-à-dire on peut déclarer et définir des fonctions et des procédures qui ne prendront aucun paramètre.

Remarque 3

- Une fonction(procédure) peut être déclarée et définie à l'intérieur d'une autre fonction(procédure ou l'algorithme principale), mais elle reste accessible qu'à l'intérieur de celle-ci, et aucune fonction(procédure) déclarée ailleurs ne peut l'utiliser.

20

Exemple : une fonction *Moyenne* déclarée, définie et appelée à l'intérieur de l'algorithme principale.

```

Algorithme Test
Variable x, y, z : Réel
Fonction Moyenne(x : Réel, y : Réel, z : Réel) : Réel
Début
  Retourner (x + y + z)/3
FinFonction
Début
  Ecrire("Donnez trois nombres")
  Lire(x,y,z)
  Ecrire("La moyenne est ", Moyenne(x,y,z))
Fin

```

- ❖ La fonction *Moyenne* n'est accessible qu'à l'intérieur de l'algorithme *Test*, elle ne peut pas être appelée par une fonction(procédure) déclarée et définie ailleurs.

21

4-Les paramètres d'une fonction(Procédure)

4.1-Paramètre formels et paramètres effectifs**4.1.1-Paramètre formels**

- ❖ Les noms des paramètres figurant dans la déclaration d'une fonction(ou procédure) sont appelés **paramètres ou arguments formels**.
- ❖ À l'aide des paramètres formels on peut décrire le comportement de la fonction en manipulant ceux-ci dans son corps.
- ❖ Par exemple, dans la déclaration de la fonction Factoriel :

```

Fonction Factoriel(n : Entier)
.
.
FinFonction

```

Le paramètre *n* est un paramètre formel

22

4.1.2- Paramètre effectifs

- ❖ Les paramètres fournis à une fonction lors de son appel sont appelés **paramètres ou arguments effectifs**.
- ❖ À l'aide des paramètres effectifs, la fonction peut réaliser le traitement décrit dans son corps.
- ❖ Par exemple, dans la déclaration de la fonction Factoriel :

```

.
  Ecrire("n!=",Factoriel(n), "m!=", Factoriel(m))
.

```

Les paramètres *n* et *m* sont des paramètres effectifs

- ❖ Lors d'un appel à une fonction, les paramètres formels sont remplacés par les paramètres effectifs.

23

- ❖ Un paramètre effectif peut être une expression quelconque qui retourne une valeur de même type que le paramètre formel correspondant.

- ❖ Dans ce cas, le paramètre formel peut être une valeur, une variable, une fonction, une expression arithmétique, etc.

Exemple

La fonction factoriel peut être appelée par les différentes manières suivantes :

- Factoriel(24), Factoriel(2*10)
- Factoriel(k), Factoriel(2*(k-p))
- Factoriel(Ent(x))
- Factoriel(Factoriel(n))

24

4.2- Passage des paramètres

4.2.1- Passage par valeur

- ❖ Le passage par valeur consiste à passer à la fonction(ou procédure) des copies des paramètres effectifs et non les paramètres effectifs eux même.
- ❖ Ce type de passage présente à la fois un avantage et un inconvénient :
 - L'avantage est qu'à l'intérieur de la fonction on travaille qu'avec une copie des données et les données originelles restent intactes.
 - Cette avantage devient un inconvénient si on veut que la fonction intervienne sur les données originelles comme par exemple la modification ou la suppression.
- ❖ Le passage par valeur est déconseillé quand il s'agit des données complexes (voir plus loin).

25

Exemple

```

Procédure Incrémenter(n : Entier)
Début
  n ← n + 1
  Ecrire("Dans la procédure : n = ", n)
FinProcédure
Algorithme Passage_Par_Valeur
Variable n : Entier
Début
  n ← 10
  Ecrire("n = ", n)
  Incrémenter(n)
  Ecrire("n = ", n)
Fin
Cet algorithme affiche :
  n = 10
  Dans la fonction : n = 11
  n = 10

```

26

4.2.2- Passage par adresse ou par référence

- ❖ Le passage par adresse consiste à passer à la fonction(ou procédure) les adresses des paramètres effectifs.
- ❖ Ce type de passage permet à la fonction de manipuler les paramètres effectifs à travers leurs adresses mémoire.
- ❖ En utilisant ce type de passage, les modifications apportées aux paramètres effectifs dans le corps de la fonction prendront effet même en dehors de la fonction (ou la procédure)
- ❖ Pour utiliser le passage par adresse, on utilise le mot clé Référence suivi du nom du paramètre formel et son type.

Syntaxe :

NomFonction (Référence Param : Type, ...) : Type

NomProcédure (Référence Param : Type, ...)

- ❖ Maintenant, on peut résoudre le problème rencontré dans l'exemple précédent :

27

Exemple

```

Procédure Incrémenter(Référence n : Entier)
Début
  n ← n + 1
  Ecrire("Dans la fonction : n = ", n)
FinProcédure
Algorithme Passage_Par_Référence
Variable n : Entier
Début
  n ← 10
  Ecrire("n = ", n)
  Incrémenter(n)
  Ecrire("n = ", n)
Fin
Cet algorithme affiche :
  n = 10
  Dans la fonction : n = 11
  n = 11

```

28

4.2.3- Passage des paramètres de type tableau

- ❖ Le passage des tableaux est toujours par adresse quelque soit le type de passage utilisé dans la déclaration de la fonction(ou procédure).

□ Cas des tableaux statiques :

Syntaxe :

Procédure NomProcédure (Tableau NomTab[N] : Type, ...)

Procédure NomProcédure (Tableau NomMatrice[N,M] : Type, ...)

Exemple:

Ecrivez une procédure **AfficherTableau** qui affiche un tableau de type réel passé en paramètre. La fonction prend aussi en paramètre la taille du tableau.

29

Solution

```

Procédure AfficherTableau(Tableau T[10] : Réel)
Variable i : Entier

```

Début

```

  Pour i allant de 0 Jusqu'à 9 Faire
    Ecrire(T[i])

```

```

  FinPour

```

FinProcédure

- ❖ Le problème dans cet algorithme c'est qu'il ne permet d'afficher que des tableaux de taille 10.
- ❖ La solution s'est de passer un tableau de taille variable.

Syntaxe:

Procédure NomProcédure (Tableau NomTab[] : Type, ...)

Procédure NomProcédure (Tableau NomMatrice[,] : Type, ...)

30

Exemple:

```

Procédure AfficherTableau(Tableau T[] : Réel, N:
Entier)
Variable i : Entier
Début
  Pour i allant de 0 Jusqu'à N-1 Faire
    Ecrire(T[i])
  FinPour
FinProcédure

```

□ Cas des tableaux dynamiques :

```

Procédure NomProcédure(NomTab : ^Type,...)
Procédure NomProcédure(NomMatrice : ^^Type,...)

```

- ❖ Dans ce cas on doit passer aussi la taille du tableau.
- ❖ Dans le corps de la fonction(ou procédures) on manipule le nom du pointeur comme un simple tableau.

31

5-Les variables locales et les variables globales

5.1- Variable locale

- ❖ Les variables déclarées à l'intérieur d'une fonction ou d'une procédure sont visibles qu'à l'intérieur de celles-ci.
- ❖ C'est-à-dire dès que l'exécution d'une fonction(ou procédure) est terminée, toutes ses variables locales seront détruites.

5.2- Variable globale

- ❖ Les variables globales sont des variables déclarées en dehors de toutes fonctions(procédure) et même en dehors de l'algorithme(programme)principale.
- ❖ Une variable globale est accessible à toutes les fonctions(procédure), c'est pour cela qu'elle doit être déclarée en premier, avant toute autres déclaration.

32

Exemple

```

Variable x : Réel
Procédure prc()
Variable a : Entier
Début
.
.
FinProcédure
Algorithme Test
Variable n : Entier
Début
.
.
Fin

```

x est une variable globale
accessible dans la procédure
prc() et dans l'algorithme **Test**

a est une variable local
accessible que dans la
procédure **prc()**

n est une variable local
accessible que dans
l'algorithme **Test**

33

- ❖ D'une manière générale, on a :

[Déclaration des constantes et des variables globales]

[Déclaration et définition des fonctions et des procédures globales]

Algorithme NomAlgorithme

[Déclaration des constantes et des variables locales]

[Déclaration et définition des fonctions et des procédures locales]

Début

[**Traitement**]

Fin

34