



# ALGO2

Département Mathématiques et Informatique

SMI / S3

AU 2020 - 2021

*Achtaich Khadija*

# ALGO2

## Objectives



*Achtaich Khadija*

# ALGO 2



## Plan du cours



**01 Chapitre 1**  
Introduction et rappels

---

**02 Chapitre 2**  
Les tableaux

---

**03 Chapitre 3**  
Les procédures et les fonctions

---

**04 Chapitre 4**  
La récursivité

---

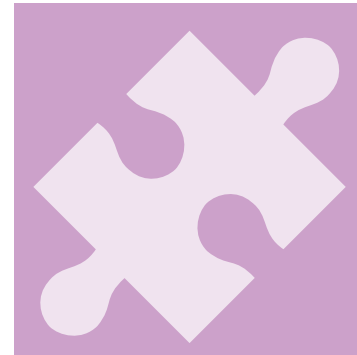
**05 Chapitre 5**  
Les enregistrements et les fichiers

---

**06 Chapitre 6**  
La complexité

# ALGO 2

## CHAPITRE 3



Les procédures et les  
fonctions

*Achtaich Khadija*

# Sous programmes

- Définition

**Un sous-programme est un traitement particulier appelé à s'exécuter à l'intérieur d'un autre programme**

- Utilité :

- **quand un même traitement doit être réalisé plusieurs fois** dans un programme. On écrit un sous-programme pour ce traitement et on l'appelle à chaque endroit où l'on en a besoin
- **pour organiser le code** , améliorer la conception et la lisibilité des gros programmes .

*Achtaich Khadija*



# Fonctions et procédures

- 2 sortes de sous-programmes :
  - les **fonctions**
  - les **procédures**
- L'appel d'une fonction est une expression, tandis que l'appel d'une procédure est une instruction :
  - une **fonction renvoie un résultat**
  - une **procédure ne renvoie rien**

*Achtaich Khadija*



# Fonctions et procédures

- 2 sortes de sous-programmes :
  - les **fonctions**
  - les **procédures**
- L'appel d'une fonction est une expression, tandis que l'appel d'une procédure est une instruction :
  - une **fonction renvoie un résultat**
  - une **procédure ne renvoie rien**

*Achtaich Khadija*



# L'utilisation des procédures et des fonctions

L'utilisation d'une fonction ou d'une procédure nécessite trois parties :

**le prototype** : c'est la déclaration nécessaire qui se place avant la fonction principale;

**l'appel** : c'est l'utilisation d'une fonction à l'intérieur d'une autre fonction (par exemple le programme principal) ;

**la déclaration** : c'est l'écriture proprement dite de la fonction ou procédure, en-tête et corps.

*Achtaich Khadija*





# Procédures

- Une procédure est un ensemble d'instructions regroupées sous un nom, qui réalise un traitement particulier dans un programme lorsqu'on l'appelle
- Une procédure est un sous-programme qui exécute un certain nombre d'actions sans fournir de valeur de retour après son exécution.

*Achtaich Khadija*



# Procédures

- Comme un programme, une procédure possède
  - un nom
  - des variables
  - des instructions
  - un début
  - une fin
- Tout comme un programme, l'écriture d'une procédure requiert un en-tête et un corps

*Achtaich Khadija*



# Définition d'une procédure

Définir une procédure consiste à :

- Définir l'**ENTETE de la procédure**
  - a. IDENTIFIER LA PROCEDURE
  - b. DECLARER LA LISTE DES PARAMETRES
- Définir le **CORPS de la procédure** :
  - a. DECRIRE L'ALGORITHME : constantes et variables, actions

*Achtaich Khadija*



# Syntaxe d'une procédure sans paramètres

**Syntaxe :**

**Procédure** nomProcédure( )

En-tête

**Var** var1 : type,...,varN: type

**Début**

instruction 1

instruction 2

instruction 3

.

.

.

instruction n

**FinProcédure**

Corps



# Appel d'une procédure sans paramètres

Pour déclencher l'exécution d'une procédure dans un programme, il suffit de l'appeler :

indiquer son nom suivi de parenthèses

```
ALGORITHME test  
VAR Entier1,Entier2 : entier
```

```
DEBUT  
instructions  
nomProcédure()  
instructions  
FIN
```

*Achtaich Khadija*



# Exemple

```
PROCEDURE Afficherchoix ()  
DEBUT
```

```
    ECRIRE("choix 1 : calculer la surface d'un rectangle")  
    ECRIRE("choix 2 : calculer la surface d'un cercle")  
    ECRIRE("choix 3 : quitter")
```

```
FINPROCEDURE
```

*Achtaich Khadija*



# Syntaxe d'une procédure avec paramètres

**Syntaxe :**

**Procédure** nomProcédure(paramètres:type) En-tête

**Var** var1 : type,...,varN: type

**Début**

instruction 1

instruction 2

instruction 3

.

.

.

instruction n

**FinProcédure**

Corps

# Appel d'une procédure avec paramètres

Pour appeler une procédure avec un paramètre, il faut mettre la valeur du paramètre entre parenthèses

```
ALGORITHME monAlgo  
VAR          Entier1 : entier  
              Chaîne : chaîne
```

```
DEBUT  
instructions  
nomProcédure(Entier1)  
nomProcédure(3)  
instructions  
FIN
```

*Achtaich Khadija*





# Exemple

$x = \text{AfficherSomme}$

## **DEFINITION**

```
PROCEDURE AfficherSomme( pNb1 : ENTIER, pNb2 : ENTIER)
DEBUT
    ECRIRE("la somme est ", (pNb1 + pNb2) )
finPROCEDURE
```

## **Appel**

```
...
LIRE(n1, n2)
→ AfficherSomme(n1, n2)
..
```

*Achtaich Khadija*



# Les fonctions - Définition

- Une fonction est un sous-programme retournant un et un seul résultat au programme appelant
- Les fonctions sont appelées pour récupérer une valeur (alors que les procédures ne renvoient aucune valeur)
- L'appel des fonctions est différent de l'appel des procédures :

L'appel d'une fonction doit **obligatoirement** se trouver à l'intérieur d'une instruction qui utilise sa valeur

Le résultat d'une fonction doit obligatoirement être retourné au programme appelant par l'instruction Retourne

*Achtaich Khadija*



# Les fonctions - Syntaxe

**Fonction** nomFonction(par1:type,...,parN:type):type *En-tête*

//déclaration des variables

**Var** Valeur : type

**Début**

instructions/\*par exemple valeur    par1+par2 \*/ Corps

**Retourne** valeur

/\*ou bien **Retourne** par1+par2 \*/

**FinFonction**

**Remarque:**

valeur est le nom d'une variable ou constante, ou correspond à une expression

# Les fonctions - Exemple

## Prototype et définition:

**FONCTION** CalculerDuree (AnDeb : ENTIER, AnFin : ENTIER) : ENTIER  
**VAR** duree : ENTIER

**DEBUT**

duree ← AnFin – AnDeb

**RETOURNE** duree

**FINFONCTION**

## Appel:

...  
LIRE(anNais, anCour)  
ECRIRE("vous avez ", CalculerDuree(anNais,anCour), "  
ans")  
...

*Achtaich Khadija*



# Variables locales/ Variables globales

- Les variables déclarées dans un programme.
- On peut alors qualifier la variable ou la constante de GLOBALE afin de signifier qu'elle devient accessible directement par les sous-programmes, sans nécessité de la passer en paramètres.
- **CETTE PRATIQUE EST CEPENDANT DECONSEILLEE : ELLE PEUT CONDUIRE A DES EFFETS NON CONTROLES (dits « effets de bord »).**

*Achtaich Khadija*



## Exercices d'application

Soit une matrice carrée de taille 50. Ecrire  
l'algorithme qui permet de faire la somme  
de la diagonale de cette matrice

*Achtaich Khadija*



# Exercice d'application

Type mat=tableau (50,50):entier

## Procédure lecture ( L :mat)

**Debut**

Pour i ← 0 à 49

Pour j ← 0 à 49

Ecrire ("L (", i, j, " ")")

Lire (L(i,j))

Fin Pour

**Fin**

## Fonction somme (A : mat) :entier

Variable D : entier

**Debut**

D ← 0

Pour i ← 0 à 49 Faire

D ← D+A(i,i)

Fin Pour

Retourne D

**Fin**

*Achtaich Khadija*



# Exercices d'application

## Algorithme somme\_diagonale

Var i, j, :entier

L :mat

**Début**

Lecture (L)

Ecrire (" la somme de la diagonale de la matrice est ", somme (L))

**Fin**

*Achtaich Khadija*





# Paramètres Et Arguments

## Définition

- Un paramètre est une variable particulière qui sert à la communication entre programme appelant et sous-programme et qui a un nom et un type

*Achtaich Khadija*



# Paramètres Et Arguments

Il est primordial de bien distinguer entre:

- Les paramètres formels qui se trouvent dans l'en-tête d'une procédure ou fonction lors de sa définition
- Les paramètres effectifs ou paramètres réels (ou arguments) désignent les valeurs réellement fournies lors de l'appel du sous-programme et qui sont placés entre parenthèses lors de l'appel

*Achtaich Khadija*



# Paramètres formels

- placés dans la **définition** d'une procédure
- Servent à **décrire** le traitement à réaliser par la procédure indépendamment des valeurs traitées
- Ce sont des variables locales à la procédure
- Ils sont déclarés dans l'entête de la procédure

*Achtaich Khadija*



# Paramètres effectifs

- placés dans **l'appel** d'une procédure
- Lors de l'appel, leurs valeurs sont transmises aux paramètres formels correspondants
- Un paramètre effectif en donnée peut être  
soit une variable du programme appelant  
soit une valeur littérale  
soit le résultat d'une expression

*Achtaich Khadija*



# Modes de passage

- Les modes de passage des paramètres définissent comment les valeurs sont passées de le programme principal (ou d'un sous-programme) au sous-programme appelé.
- Le mode de passage est précisé à la déclaration de chacun des paramètres.

*Achtaich Khadija*



# Passage par valeur

- La valeur du paramètre effectif est copiée dans le paramètre formel. Le paramètre effectif ne subit aucun changement.
- **Le passage par valeur est l'option par défaut dans la déclaration des paramètres.**

*Achtaich Khadija*



# Passage par Référence

- toute modification sur le paramètre formel modifie le paramètre effectif. Ce dernier reçoit le résultat.
- Dans le **PASSAGE PAR REFERENCE**, le **PARAMETRE FORMEL REÇOIT UNE REFERENCE VERS LE CONTENU DU PARAMETRE Effectif** : il devient ainsi comme un **alias d'un paramètre réel**.
- Toute action effectuée sur le paramètre formel est ainsi reportée directement sur la valeur du paramètre réel.
- Dans ce cas, **LES VALEURS D'ORIGINE PEUVENT ETRE MODIFIEES PAR LES INSTRUCTIONS DU SOUS-PROGRAMME** (sauf si l'argument est défini comme **CONST** au lieu de **VAR**).

*Achtaich Khadija*



# Modes de Passage: Exemples

Soit la fonction suivante:

Fonction incremente( i:entier):

Début

i ← i+1

retourne i

Fin fonction

...et on va appeler la  
fonction incremente

Algorithme TestPassageValeur

Var i, j:entier

Début

i ← 0

j ← incremente(i);

Ecrire("i=" , i);

Ecrire("j=" , j);

Fin

Quelles seront les  
valeurs affichés par  
l'algorithme pour i et j  
après exécution?

*Achtaich Khadija*





# Modes de Passage: Exemples

Quand nous exécutons TestPassageValeur

i=0

j=1

... donc la valeur de i n'a pas été modifiée !

*Achtaich Khadija*



# Modes de Passage: Exemples

On modifie la fonction comme suit:  
Fonction incremente( Par reference

i:entier):

Début

i ← i+1

retourne i

Fin fonction

...et on va appeler la  
fonction incremente

Algorithme TestPassageReference

Var i, j:entier

Début

i ← 0

j ← incremente(i);

Ecrire("i=" , i);

Ecrire("j=" , j);

Fin

Quelles seront les  
valeurs affichés par  
l'algorithme pour i et j  
après exécution?

# Modes de Passage: Exemples

Quand nous exécutons TestPassageReference

i=1

j=1

... donc la valeur de i a été modifiée !

*Achtaich Khadija*

