



مدرسة علوم المعلومات
+٩١٢١ | +٢٥.٥٥.٥١ | ٤١٢٤٤٥١
ECOLE DES SCIENCES
DE L'INFORMATION
www.esi.ac.ma

Algorithmique

2. Les variables

Sommaire

- I. Définitions et généralités
- II. Déclaration et affectation
- III. Expressions et opérateurs
- IV. Lecture / Écriture
- V. Commentaires

Rappel

Rappel

- ▶ Algorithmique

- ▶ Discipline qui a pour objet la conception, l'évaluation et l'optimisation des méthodes de calcul en mathématiques et en informatique

Rappel

- ▶ **Algorithme**

- ▶ Suite d'instructions qui, si elles sont exécutées correctement, permettent de résoudre un problème donné

Rappel

- ▶ Informatique

- ▶ Science du traitement automatique et rationnel de l'information considérée comme le support des connaissances et des communications

Rappel

- ▶ Algorithme informatique

- ▶ Suite d'instructions ordonnée qui décrit de façon exhaustive les différentes étapes à suivre par un processeur pour résoudre un problème donné en un temps fini

Rappel

- ▶ L'ordinateur n'est pas capable d'initiative mais peut stocker des programmes et les exécuter
- ▶ Programme informatique
 - ▶ Implémentation d'un ou plusieurs **algorithme(s)** dans un **langage de programmation** donné

Rappel

**Problème dans le
monde réel**

Analyse

Algorithme

Codage avec un langage de programmation

Programme

Exécution

Résultat

Rappel

- ▶ Langage de programme
 - ▶ Notation conventionnelle destinée à implémenter des algorithmes afin de produire des programmes informatiques
 - ▶ Langages de bas niveau : machine, assembleur
 - ▶ Langages de haut niveau : C Java Python ...

Rappel

- ▶ Un algorithme est destiné à des êtres humains qui vont généralement l'implémenter dans un langage de programmation quelconque
- ▶ Il est indépendant du langage de programmation
- ▶ Son écriture doit détailler le plus possible son fonctionnement et sa structure

Rappel

- ▶ Pseudo-code algorithmique
 - ▶ Formalisation de l'écriture des algorithmes dans une langue humaine (généralement l'anglais) en adoptant quelques conventions proche des langages de programmation mais sans les contraintes syntaxiques de ces derniers

Rappel

► Pseudo-code algorithmique

```
ALGORITHME nom_de_l'algorithme  
<Déclarations>  
DEBUT  
<Instructions>  
//commentaire  
FIN
```

Définitions et généralités

Variable

- ▶ Une **variable** est une cellule mémoire désignée par un identificateur et possédant un contenu qui est consultable et modifiable par des programmes
 - ▶ Est d'un type donné
 - ▶ A une valeur
 - ▶ Doit être initialisée avant d'être consultée

Variable

- ▶ Une **variable** est une cellule mémoire désignée par un identificateur et possédant un contenu qui est consultable et modifiable par des programmes
- ▶ Elle sert à stocker les résultats des calculs intermédiaires et finaux réalisés dans l'algorithme

Identificateur

- ▶ Un **identificateur** est le nom associé à une variable, un type ou un sous-programme
- ▶ Composé d'une suite de lettre et de chiffres
 - ▶ Doit commencer par une lettre
 - ▶ Interdits: espaces, lettres accentuées ou avec cédille, caractères spéciaux sauf "_"
- ▶ Sensible à la casse (ex: var \neq VAR \neq Var)

Identificateur

- ▶ Un **identificateur** est le nom associé à une variable, un type ou un sous-programme
- ▶ Par convention
 - ▶ Utilisation de la notation camel case pour les variables (ex: `variableDuProgramme`, `nomEtudiant`, `longueurRectangle...`)
 - ▶ Utilisation des majuscules pour les constantes

Identificateur

- ▶ Un **identificateur** est le nom associé à une variable, un type ou un sous-programme
- ▶ Il est recommandé de choisir un identificateur qui soit le plus significatif et compréhensible possible
 - ▶ Ex: nomEtudiant au lieu de n ou x ou var ...

Identificateur

- ▶ Exemples d'identificateurs valides
 - ▶ A , b , X ...
 - ▶ T5 , w3c ...
 - ▶ longueur , etudiant
 - ▶ longueurDuRectangle , nomEtudiant
 - ▶ longueur_du_rectangle , nom_etudiant
 - ▶ ...

Identificateur

- ▶ Exemples d'identificateurs non valides
 - ▶ 5G , t&st
 - ▶ longueur du rectangle , nom Etudiant
 - ▶ longueur-du-rectangle , nom-etudiant
 - ▶ nom_étudiant , remplaçant
 - ▶ ...

Type

- ▶ Le **type** d'une variable définit l'ensemble des valeurs qu'elle peut avoir
- ▶ Permet de traduire les valeurs de leur représentation binaire vers une représentation adaptée à la programmation dans un langage de haut niveau
- ▶ Peut être prédéfini ou défini par l'utilisateur

Type

- ▶ Les **types** prédéfinis sont
 - ▶ Entier : stocker les nombres entiers signés (4 octets)
 - ▶ Réel : stocker les nombres à virgule (8 octets)
 - ▶ Booléen : stocker des valeurs binaires vrai/faux (1 octet)
 - ▶ Caractère : stocker des caractères ANSI (1 octet)
 - ▶ Chaîne: stocker une chaîne de caractères

Type

- ▶ Chaque type a un ensemble d'opérations possibles
 - ▶ Entier : $+$ $-$ $*$ $/$ DIV ...
 - ▶ Réel : $+$ $-$ $*$ $/$...
 - ▶ Booléen : ET OU ...
 - ▶ Caractères et chaînes : $<$ $>$ $=$...

Remarque

- ▶ L'**identificateur** et le **type** d'une variable sont **fixés** lors de sa déclaration et ne changent pas au cours de l'exécution du programme
- ▶ La **valeur** d'une variable **évolue** au cours de l'exécution du programme

Déclaration et affectation

Déclaration des variables

- ▶ Les variables sont déclarées en début de l'algorithme

ALGORITHME *nom_de_l'algorithme*

<Partie déclarative>

DEBUT

<Instructions>

//commentaire

FIN

<déclarations des constantes>

<déclarations des variables>

<déclarations des routines>

Déclaration des variables

- ▶ Les variables sont déclarées en début de l'algorithme
 - ▶ Syntaxe de la déclaration

VAR *identificateur* : *type*

VAR nomEtudiant : **chaîne**

VAR absent : **booléen**

VAR lettre : **caractère**

VAR longueur, largeur : **réel**

Déclaration des constantes

- ▶ Les constantes sont déclarées en début de l'algorithme
- ▶ Syntaxe de la déclaration

CONSTANTE *IDENTIFICATEUR* = *valeur*

CONSTANTE PI = 3.14

CONSTANTE TVA = 0.20

Affectation des variables

- ▶ L'affectation est l'opération qui permet d'attribuer une valeur à une variable
- ▶ Elle établit un lien entre l'identificateur et l'emplacement mémoire de la valeur correspondante
- ▶ Ce lien est réalisé par des pointeurs

Affectation des variables

- ▶ L'affectation est l'opération qui permet d'attribuer une valeur à une variable
- ▶ Syntaxe de l'affectation

ma_variable ← valeur

nomEtudiant ← "Ali"

longueur ← 10

absent ← False

largeur ← 5

lettre ← "a"

Affectation des variables

- ▶ L'affectation est l'opération qui permet d'attribuer une valeur à une variable
- ▶ Syntaxe de l'affectation

ma_variable ← valeur

nomEtudiant ← "Ali"

longueur ← 10

absent ← False

largeur ← 5

lettre ← "a"

Valeurs constantes

Affectation des variables

- ▶ L'affectation est l'opération qui permet d'attribuer une valeur à une variable
- ▶ Syntaxe de l'affectation

ma_variable ← expression

- ▶ Opérations arithmétiques et/ou logiques sur des constantes, variables et fonctions

Affectation des variables

- ▶ L'affectation est l'opération qui permet d'attribuer une valeur à une variable
- ▶ Syntaxe de l'affectation

ma_variable ← expression

perimetre ← 2*(longueur + largeur) x ← perimetre + 10

nomComplet ← nomEtudiant + " , " + prenomEtudiant

absent ← True OU False

Affectation des variables

- ▶ L'affectation est l'opération qui permet d'attribuer une valeur à une variable
- ▶ Syntaxe de l'affectation

ma_variable ← expression

perimetre ← 2*(longueur + largeur) x ← perimetre + 10

nomComplet ← nomEtudiant + " , " + prenomEtudiant

absent ← True OU False

Expressions évaluables

Affectation des variables

- ▶ L'affectation est l'opération qui permet d'attribuer une valeur à une variable
- ▶ Elle est effectuée en deux étapes
 - ▶ 1^{ère} étape : évaluation de l'expression qui est dans la partie droite de l'affectation
 - ▶ 2^e étape : stockage de la valeur obtenue dans la 1^{ère} étape dans la variable qui est dans la partie gauche de l'affectation

Affectation des variables

► Exemple

```
ALGORITHME mon_algo  
VAR A, B : entier  
DEBUT  
    A ← 2  
    B ← A + 5  
FIN
```

Expressions et opérateurs

Expressions et opérateurs

- ▶ Une **expression** est une suite d'**opérateurs** et de termes qui est compréhensible et qu'on peut calculer
 - ▶ Opérateurs arithmétiques
 - ▶ Opérateur alphanumérique
 - ▶ Opérateurs logiques
 - ▶ Opérateurs relationnels

Expressions et opérateurs

► Opérateurs arithmétiques

Opérateur	Description	Opérandes	Type du résultat
—	Soustraction Changement de signe (opérateur unaire)	Entiers ou réels	Même type que les opérandes
+	Addition	Entiers ou réels	Même type que les opérandes
*	Multiplication	Entiers ou réels	Même type que les opérandes
/	Division flottante	Entiers ou réels	Réel
DIV	Division entière	Entiers seulement	Entier
MOD	Modulo	Entiers seulement	Entier

Expressions et opérateurs

- ▶ Opérateur alphanumérique

- ▶ L'opérateur **&** peut être utilisé avec les caractères et les chaînes de caractères pour l'opération de **concaténation**

nomComplet ← nomEtudiant & " , " & prenomEtudiant

Expressions et opérateurs

► Opérateurs logiques

Opérateur	Description	Opérandes	Type du résultat
NON	Négation logique \neg	Booléens	Booléen
ET	Et logique (AND) \wedge	Booléens	Booléen
OU	Ou logique (OR) \vee	Booléens	Booléen
OUEX	Ou exclusif (XOR) $\underline{\vee}$	Booléens	Booléen

Expressions et opérateurs

► Opérateurs logiques

► Rappel

	NON
0	1
1	0

ET	0	1
0	0	0
1	0	1

OU	0	1
0	0	1
1	1	1

OUEX	0	1
0	0	1
1	1	0

Expressions et opérateurs

► Opérateurs relationnels

Opérateur	Description	Opérandes	Type du résultat
=	Égale	Types compatibles	Booléen
<>	Différent (noté aussi !=)	Types compatibles	Booléen
<	Inférieur à	Types compatibles	Booléen
>	Supérieur à	Types compatibles	Booléen
<=	Inférieur ou égal à	Types compatibles	Booléen
>=	Supérieur ou égal à	Types compatibles	Booléen

Expressions et opérateurs

- ▶ Évaluation des expressions
 - ▶ Effectuée selon la priorité des opérateurs

Opérateurs unaires	– NON
Opérateurs multiplicatifs	* / DIV MOD ET
Opérateurs additifs	+ – OU
Opérateurs relationnels	= < > ≤ ≥ <>

Expressions et opérateurs

- ▶ Évaluation des expressions
 - ▶ Effectuée selon la priorité des opérateurs
 - ▶ Les expressions entre parenthèses sont évaluées avant d'intervenir dans le reste des calculs

Expressions et opérateurs

► Évaluation des expressions

- Dans une affectation, le membre de gauche peut faire intervenir le membre de droite, comme dans le cas de **l'incréméntation**

$$i \leftarrow i + 1$$

- On commence par évaluer l'expression à droite puis on met à jour la valeur de la variable dans le membre de gauche

Exercices

- ▶ Donner les valeurs des variables après exécution de l'algorithme

```
ALGORITHME mon_algo  
VAR A, B : entier  
DEBUT  
    A ← 2  
    B ← A * 5  
FIN
```


Exercices

- ▶ Donner les valeurs des variables après exécution de l'algorithme

```
ALGORITHME mon_algo  
VAR A, B : entier  
DEBUT  
    A ← 2  
    B ← A * 5  
    A ← 0  
FIN
```

Exercices

- ▶ Donner les valeurs des variables après exécution de l'algorithme

```
ALGORITHME mon_algo  
VAR A, B : entier  
DEBUT  
  A ← 2  
  B ← A * 5  
  A ← A + 1  
  B ← A - B  
FIN
```

Exercices

- ▶ Donner les valeurs des variables après exécution de l'algorithme

```
ALGORITHME mon_algo  
VAR A, B, C: entier  
DEBUT  
  A ← 2  
  B ← A * 5  
  C ← A + B  
FIN
```

Exercices

- ▶ Donner les valeurs des variables après exécution de l'algorithme

```
ALGORITHME mon_algo  
VAR A, B, C: entier  
DEBUT  
  A ← 2  
  B ← 5  
  C ← A + B  
  B ← A + B  
  A ← C  
FIN
```

Exercices

- ▶ Donner les valeurs des variables après exécution de l'algorithme

```
ALGORITHME mon_algo  
VAR A, B, C: entier  
DEBUT  
  A ← 2  
  B ← A + 5  
  C ← A + B  
  A ← C  
  C ← A MOD B  
FIN
```

Exercices

- Donner les valeurs des variables après exécution de l'algorithme

```
ALGORITHME mon_algo  
VAR A, B: entier  
VAR C: réel  
DEBUT  
    A ← 2  
    B ← A + 5  
    C ← B / A  
FIN
```

Exercices

- ▶ Donner les valeurs des variables après exécution de l'algorithme

```
ALGORITHME mon_algo  
VAR A, B : entier  
DEBUT  
  A ← 2  
  B ← 5  
  A ← B  
  B ← A  
FIN
```

Exercices

- Donner les valeurs des variables après exécution de l'algorithme

```
ALGORITHME mon_algo  
VAR A, B : entier  
DEBUT  
  A ← 2  
  B ← 5  
  B ← A  
  A ← B  
FIN
```


Exercices

- ▶ Donner les valeurs des variables après exécution de l'algorithme

```
ALGORITHME mon_algo  
VAR A, B : chaîne  
DEBUT  
  A ← "Hello"  
  B ← "Bonjour"  
  A ← A & " " & B  
FIN
```

Exercices

- ▶ Donner les valeurs des variables après exécution de l'algorithme

```
ALGORITHME mon_algo  
VAR A, B : chaîne  
DEBUT  
  A ← "1"  
  B ← "2"  
  A ← A & B  
FIN
```

Exercices

- ▶ Que fait cet algorithme ?

```
10  if A = B goto 70
20  if A < B goto 50
30  A = A - B
40  goto 10
50  B = B - A
60  goto 10
70  end
```

Exercices

- ▶ Écrire un algorithme pour échanger les valeurs de deux variables
- ▶ Valeurs de départ : $A = 2$ et $B = 5$ (ou toute autre valeur)
- ▶ Valeurs finales : $A = 5$ et $B = 2$ (ou mêmes valeurs utilisées au départ)
 - ▶ Indication: utiliser une variable intermédiaire

Exercices

- ▶ Écrire un algorithme pour échanger les valeurs de deux variables

```
ALGORITHME mon_algo  
VAR A, B, C : entier  
DEBUT  
    A ← 2  
    B ← 5  
    C ← A  
    A ← B  
    B ← C  
FIN
```

Lecture/Écriture

Instructions

- ▶ Une instruction est une action élémentaire à accomplir par l'algorithme afin de
 - ▶ Effectuer un calcul
 - ▶ Communiquer avec un périphérique d'entrée
 - ▶ Communiquer avec un périphérique de sortie

Instructions

- ▶ On compte quatre instructions de base
 - ▶ Déclaration (mémoire)
 - ▶ Affectation (calcul)
 - ▶ Lecture (entrées)
 - ▶ Écriture (sorties)

Lecture

- ▶ Il est nécessaire de lire des valeurs saisies au clavier par l'utilisateur et de les affecter à des variables
- ▶ Instruction **Lire()** avec entre parenthèse les identificateurs des variables à saisir

Lecture

► Exemples

ALGORITHME *mon_algo*

VAR *x* : **réel**

DEBUT

Lire(*x*)

FIN

Lecture

► Exemples

ALGORITHME *calcul_du_périmètre*

VAR longueur, largeur, p : **réel**

DEBUT

Lire(longueur, largeur)

$p \leftarrow 2 * (\text{longueur} + \text{largeur})$

FIN

Écriture

- ▶ Il est nécessaire de visualiser (i.e. afficher sur l'écran) du texte ou les valeurs des variables
- ▶ Instruction **Afficher()** avec entre parenthèses
 - ▶ Les identificateurs des variables à visualiser
 - ▶ Une expression dont la valeur calculée sera affichée
 - ▶ Du texte brute entre " "

Écriture

► Exemples

ALGORITHME *mon_algo*

VAR x : réel

DEBUT

Afficher("Saisir la valeur de x")

Lire(x)

Afficher("La valeur de x est :", x)

FIN

Écriture

► Exemples

ALGORITHME *calcul_du_périmètre*

VAR longueur, largeur, p : **réel**

DEBUT

Afficher("Donner la longueur et la largeur: ")

Lire(longueur, largeur)

$p \leftarrow 2 * (\text{longueur} + \text{largeur})$

Afficher("Le périmètre est : ", p)

FIN

Exercices

- ▶ Quel est le résultat de l'algorithme suivant?

```
ALGORITHME mon_algo  
VAR A, B : entier  
DEBUT  
    A ← 100  
    B ← 100*2  
    Afficher(A)  
    Afficher(B)  
FIN
```

Exercices

- ▶ Quel est le résultat de l'algorithme suivant?

```
ALGORITHME mon_algo  
VAR A, B : entier  
DEBUT  
    Lire(A)  
    Lire(B)  
    Afficher(A*B)  
FIN
```


Exercices

- ▶ Écrire un algorithme qui demande à un utilisateur de saisir un nombre puis calcule et affiche le carré de ce nombre

Exercices

- ▶ Écrire un algorithme qui demande à un utilisateur de saisir un nombre puis calcule et affiche le carré de ce nombre

```
ALGORITHME calculCarre  
VAR x : réel  
DEBUT  
    Afficher("Saisir un nombre")  
    Lire(x)  
    Afficher(x*x)  
FIN
```

Commentaires

Commentaires

- ▶ Les commentaires sont des lignes de codes qui ne seront pas exécutées
 - ▶ Ils sont ignorés par le compilateur / interpréteur
- ▶ Ils servent à donner des indications sur le fonctionnement de l'algorithme
- ▶ Ils peuvent figurer dans n'importe quelle partie de l'algorithme (en-tête, déclarations, corps)

Commentaires

- ▶ Les commentaires sont aussi utilisés pour annuler des bouts de code de l'algorithme tout en les gardant (pour une réutilisation ultérieure, un besoin de test, ...) sans avoir à les effacer
- ▶ Ils sont très utiles quand l'algorithme devient long
- ▶ Il est vivement recommandé de les utiliser dans les algorithmes / programmes

Commentaires

- ▶ Syntaxe

- ▶ Commentaire sur une ligne

// ceci est un commentaire

- ▶ Commentaire sur plusieurs lignes

**/* ceci
est
un
Commentaire */**

Commentaires

► Exemple

```
ALGORITHME mon_algo  
VAR A, B : entier  
DEBUT  
    A ← 100  
    B ← 100*2  
    Afficher(A)  
    // Afficher(B)  
FIN
```

Commentaires

► Exemple

```
ALGORITHME mon_algo  
VAR A, B : entier  
DEBUT  
    A ← 100  
    B ← 100*2  
    Afficher(A)  
    // Afficher(B)  
FIN
```


Commentaires

► Exemple

```
ALGORITHME mon_algo  
VAR A, B : entier  
DEBUT  
    /* A ← 100  
    B ← 100*2 */  
    A ← 200  
    B ← 10*A  
    Afficher(A)  
    Afficher(B)  
FIN
```

Commentaires

► Exemple

```
ALGORITHME mon_algo  
VAR A, B : entier  
DEBUT  
    /* A ← 100  
    B ← 100*2 */  
    A ← 200  
    B ← 10*A  
    Afficher(A)  
    Afficher(B)  
FIN
```



مدرسة علوم المعلومات
+٩١٢١ | +٢٥.٥٥.٥١ | ٤١٢٤٤٥١
ECOLE DES SCIENCES
DE L'INFORMATION
www.esi.ac.ma

Algorithmique

2. Les variables