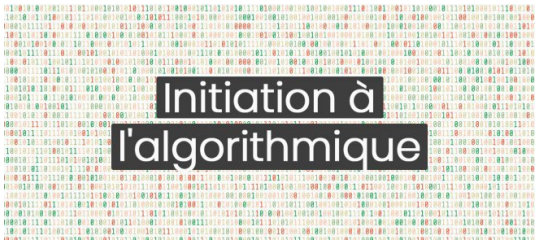
# Chapitre 1 : Éléments de base d'algorithmique

*ALGORITHMIQUE (ALSD)*




Préparé par Dr. BOUCHEBBAH Fatah

*UNIVERSITÉ DE BÉJAIA*

*FACULTÉ DES SCIENCES EXACTES*

*DÉPARTEMENT D'INFORMATIQUE*

Octobre 2022
Version 4

# Table des matières

**Objectifs** 4

I - **Pré-requis** 5

II - **Test des objectifs** 6

III - **Test des pré-requis** 8

IV - **Introduction** 10

V - **Étapes de la résolution automatique d'un problème** 11

VI - **Notion d'algorithme** 12

1. Structure d'un algorithme ..... 12

VII - **Exercice** 14

VIII - **Types de données simples d'algorithmique** 15

1. Le type Entier (Int pour integer) ..... 15

2. Le type Réel (Float ou Double) ..... 15

3. Le type Caractère (Char) ..... 15

4. Le type Chaîne de caractères (Char) ..... 15

5. Le type Booléen (Bool) ..... 16

IX - **Formes de données simples d'algorithmique** 17

1. Présentation d'une constante ..... 17

2. Présentation d'une variable ..... 17

3. Présentation d'une expression ..... 19

*3.1. Expression arithmétique ..... 19*
*3.2. Expression booléenne ..... 19*
*3.3. Les opérations permises sur les types de bases ..... 19*
*3.4. Évaluation des expressions ..... 20*

X - **Les instructions de base d'un algorithme** 21

1. L'instruction LIRE (scanf) ..... 21

2. L'instruction ÉCRIRE (printf) ..... 22

3. L'instruction d'affectation ..... 22

XI - **Exercice** 25

XII - **La trace d'exécution d'un algorithme** 26

XIII - **Exercice** 28

1. Exercice : Choisissez la bonne réponse. ..... 29

2. Exercice : Cochez la (les) bonne(s) réponse(s). ..... 30

3. Exercice : Cochez la (les) bonne(s) réponse(s). ..... 31

XIV - **Test** 32

**Solutions des exercices** 34

**Références** 39

# Objectifs

Le cours "Algorithmique et Structures de Données " vise à :

- Définir les différents concepts relatifs à l'algorithmique.
- Comprendre les principes de fonctionnement des diverses structures de traitements.
- Appliquer les différents types d'instructions à des entrées pour la production des sorties attendues.

A l'issue de ce cours, l'apprenant ou l'étudiant sera capable de :

- Connaître les spécificités et les formes des données existantes.
- Différencier entre les instructions conditionnelles et les structures répétitives.
- Utiliser les diverses instructions de contrôle et structures de données pour la conception d'un algorithme.

A l'issu de ce chapitre, vous serez capable de :

- Identifier une donnée.
- Expliquer le rôle de chaque instruction de base.
- Intégrer des entrées simples et des opérations primitives dans un algorithme.

# Pré-requis

Pour pouvoir suivre le cours "Algorithmique et Structures de Données ", Il est recommandé aux apprenants ou étudiants de connaître :

- La différence entre les composants softwares et hardwares d'un ordinateur.
- Les unités d'entrée/sortie et de traitement d'un ordinateur.
- Les fonctions de base d'un système d'exploitation.

# Test des objectifs

II

Exercice

*[solution n°1 p.34]*

En algorithmique, toutes les données à traiter sont d'une même nature.

- Vrai
- Faux

Exercice

*[solution n°2 p.34]*

L'instruction qui permet de lire une donnée simple à partir du clavier est l'instruction SI ... SINON.

- Vrai
- Faux

Exercice

*[solution n°3 p.34]*

L'instruction LIRE est une instruction d'opération.

- Vrai
- Faux

Exercice

*[solution n°4 p.34]*

Cochez la (les) bonne(s) réponse(s) si elle(s) existe(nt).

- ☐ Un algorithme présente une solution automatique à un problème.
- ☐ Un algorithme est constitué de variables uniquement.
- ☐ Un algorithme peut s'exécuter sur un ordinateur.
- ☐ On peut réaliser un algorithme sans comprendre le problème à résoudre.
- ☐ Un algorithme contient souvent plusieurs type d'instructions.
- ☐ On ne peut pas faire de la programmation sans se disposer d'un ordinateur.

# Test des pré-requis

Exercice

*[solution n°5 p.35]*

L'unité central est un composant software.

- O Vrai
- O Faux

Exercice : Cochez la (les) bonne(s) réponse(s) si elle(s) existe(nt).

*[solution n°6 p.35]*

Le clavier d'un ordinateur :

- ☐ Est un composant hardware d'un ordinateur.
- ☐ Est un composant software d'un ordinateur
- ☐ Fait partie des unités d'entrée/sortie d'un ordinateur.
- ☐ Fait partie des unités de traitement d'un ordinateur.

Exercice

*[solution n°7 p.35]*

Définissez brièvement un système d'exploitation.

Exercice

*[solution n°8 p.35]*

Réel (Real) | Expression | Caractère (Char) | Constante | Chaîne de caractères (String)

Booléen (Boolean) | Entier (Integer) | Variable | Affectation | LIRE (READ)

ÉCRIRE (WRITE)

| Types de données | Formes de données | Instructions |
|---|---|---|
| | | |

# Introduction

IV

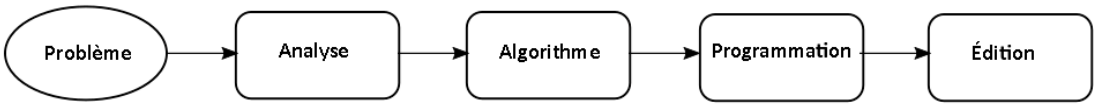
*L'algorithmique* est la science de la conception et de l'étude des algorithmes. Concevoir un algorithme de résolution d'un problème est synonyme à la proposition d'une méthode de résolution automatique du dit problème. De ce fait, l'algorithmique est considérée comme l'un des éléments essentiels de l'informatique, car le traitement automatique de l'information ne pourra pas exister sans l'algorithmique.

Dans ce chapitre, nous exhibons les éléments de base d'algorithmique, principalement, la notion d'algorithme, les types et les formes de données simples utilisées dans un algorithme, ainsi que les instructions primitives qui permettent de manipuler ces données.

V

# Étapes de la résolution automatique d'un problème

En informatique, la résolution automatique d'un problème s'effectue en suivant les 4 étapesp.39 illustrées dans la figure suivante :




*Étapes essentielles de la résolution automatique d'un problème.*

D'une manière générale, l'*étape Analyse*, sert comme une étape d'étude dans laquelle le problème soulevé est analysé dans le but de sa compréhension. Ceci est fait souvent en répondant à 3 questions :

- *Question 1 :* Qu'est ce que nous avons comme données à traiter (les entrées)?
- *Question 2 :* Que devions nous avoir après le traitement des données (Les résultats ou sorties) ?
- *Question 3 :* Que devions nous faire aux données pour avoir les résultats voulus ?

Après cette étape d'analyse, on passe à l'*étape Algorithme* dans laquelle nous écrivons un algorithme qui présente une solution au problème soulevé. Une fois l'algorithme est finalisé, nous le traduisons, dans l'*étape Programmation*, en un programme exécutable sur ordinateur. Cette traduction est faite en utilisant un langage spécifique que la machine comprend appelé langage de programmation. Finalement, dans l'*étape Édition*, nous introduisons le programme dans l'ordinateur pour voir les résultats de son exécution sur des données réelles du problème.

# Notion d'algorithme

VI

L'appellation *algorithme* est inspirée du nom du mathématicien musulman ouzbèk *Abou Jaafar Mohamed Ibn Moussa Al Khawarizmi*[p.39] . Un algorithme est une liste (ou ensemble) ordonnée d'opérations primitives, appelées *instructions*, réalisant un traitement spécifique pour la résolution d'un problème[p.39] .

### *Définition : Instruction*

Une *instruction* est une action appliquant une opération définie sur une ou plusieurs données[p.39] . En algorithmique, nous trouvons diverse formes et types de données. De ce fait, différents types d'instructions doivent exister aussi pour traiter ces données[p.39] .

## 1. Structure d'un algorithme

Un algorithme de base comprend généralement 4 parties, et sa syntaxe globale est la suivante :

### *Syntaxe : Structure d'un algorithme*

```
1 ALGORITHME Algo1; {(1) En-tête l'algorithme}
2 CONSTANTE
3       {(2) Déclaration des constantes}
4 VARIABLE
5       {(3) Déclaration des variables}
6 DÉBUT
7       {(4) Le corps de l'algorithme}
8       {On trouve ici la liste des instructions de l'algorithme}
9 FIN.
```

Les parties énumérées de l'algorithme ci-dessus sont décrites comme suit :

*(1)* L'en-tête l'algorithme : commence par le mot *Algorithme*, suivie du *nom de l'algorithme*.

*(2)* La définition des constantes de l'algorithme : commence par le mot clé CONSTANTE (ou CONST).

*(3)* La définition des variables de l'algorithme : commence par le mot clé VARIABLE (ou VAR)

*(4)* Le corps de l'algorithme : cette partie est comprise entre deux mots clés : *Début* et *Fin*. Le corps de l'algorithme contient l'ensemble des instructions, qui sont écrites selon un ordre bien définie. Chaque instruction se termine par un caractère spécial, le point virgule (;). Les instructions d'un algorithme sont exécutées séquentiellement. Explicitement, en commençant par l'instruction qui vient juste après le mot clé *Début* jusqu'à la dernière instruction qui précède le mot clé *Fin*.

*Remarque : Langage C*

La structure d'un programme en langage C est légèrement différente de celle d'un algorithme. De plus, il faut traduire les mots clés utilisés dans l'algorithmique, qui sont en français, à l'anglais.

```
#INCLUDE <STDIO.H>  (1) En-tête l'algorithme
VOID MAIN ()
{
  (2) Le corps du programme. On trouve ici la déclaration des constantes et des variables, ainsi que la liste des instructions de programme
}
```

# Exercice

VII

*[solution n°9 p.36]*

Un algorithme se construit :

- Après l'écriture d'un programme.
- Avant l'écriture d'un programme.
- L'ordre de construction n'a pas d'importance.

# Types de données simples d'algorithmique

VIII

Un *type* détermine l'ensemble des valeurs que peut prendre une donnée[p.39] . Un type est caractérisé par *ses valeur*s et *les opérations* que nous pouvons effectuer sur les données ayant ce type. Les types des données simples d'algorithmique sont : *Entier*, *Réel*, *Caractère*, *Chaîne de caractères*, et *Booléen*.

## 1. Le type Entier (Int pour integer)

Une donnée de type Entier désigne une quantité indivisible (en mathématiques : Z, l'ensemble des entiers)[p.39] . Un entier est représenté sur 16 bits (2 octets).

## 2. Le type Réel (Float ou Double)

Une donnée est réelle si sa valeur est fractionnaire (en mathématiques: R, l'ensemble des réels)[p.39] . Un réel est représenté sur 4 ou 6 octets.

## 3. Le type Caractère (Char)

Une donnée de type Caractère peut avoir tous les symboles que nous trouvons sur le clavier d'un ordinateur : espace, lettres, chiffres, signes de ponctuation et caractères spéciaux (#, @, &, etc.)[p.39] . Un caractère est représenté sur un octet.

*Remarque : Caractère et nom d'une donnée*

Une donnée de type Caractère doit être entre apostrophes. Ainsi, dans un algorithme :

A : désigne un nom d'une donnée.

'A' : désigne une valeur (la lettre A majuscule) d'une donnée.

## 4. Le type Chaîne de caractères (Char)

Une donnée de type Chaîne caractères est une suite de caractères[p.39] . Le nombre maximum de

caractères d'une chaîne caractères est 255. Chaque caractère est codé sur 1 octet.

*Exemple : Chaîne de caractères*

'LICENCE INFORMATIQUE' est une chaîne de caractères, elle est représentée sur 20 octets. Le blanc (espace) est un caractère.

## 5. Le type Booléen (Bool)

Une donnée de type Booléen représente uniquement deux valeurs logiques : *VRAI* ou *FAUX*.

# Formes de données simples d'algorithmique

IX

Une donnée (simple) à traiter par les instructions d'un algorithme peut être : une *constante*, une *variable* , ou une *expression*[p.39] .

## 1. Présentation d'une constante

Une constante est la forme  d'une donnée caractérisée par un *nom fixe*, un *type fixe*, et une *valeur fixe* [p.39] . La valeur d'une constante, une fois déclarée dans un algorithme, *ne peut en aucun cas être modifiée lors du déroulement de l'algorithme*. De ce fait, elle est conservée jusqu'à la terminaison de son déroulement.

### *Exemple : La constante Pi*

La constante Pi = 3,14 est caractérisée par :

- Nom : Pi.
- Type : Réel.
- Valeur : 3,14.

### *Syntaxe : Déclaration d'une constante.*

Une constante d'un algorithme est déclarée dans la partie (2) de l'algorithme, par la manière suivante [p.39] :

```
1 CONSTANTE(ou CONST) identificateur = Valeur ;
```

En langage C :

```
1  int identificateur = Valeur ;
```

## 2. Présentation d'une variable

Contrairement à une constante, une variable est la forme d'une donnée *dont la valeur peut être redéfinie à tout moment lors du déroulement de l'algorithme* dans lequel elle est déclarée. Donc, elle est

caractérisée par un *nom fixe*, un *type fixe*, mais par une *valeur modifiable*[p.39] .

### *Exemple : La note d'un examen*

La note d'un examen peut être vue comme une variable qui peut avoir différentes valeurs possibles comprises entre 0 et 20 :

- Nom : Note.
- Type : Réel.
- Valeur : 12, 10.25, 18.5, etc.

### *Syntaxe : Déclaration d'une variable*

Les variables d'un algorithme sont déclarées dans la partie (3) de l'algorithme, par la manière suivante [p.39] :

```
VARIABLE (ou VAR)  liste d'identificateurs : TYPE ;
```

En langage C :

```
TYPE liste d'identificateurs ;
```

### *Exemple : Déclaration de quelques variables*

```
VARIABLE (ou VAR)  Age, Som, Nombre : Entier ;
                   Prix, Moyenne : Réel;
                   Chiffre: Caractère;
                   Chaine : Chaine de caractère ;
                   Gagner : Booléen ;
```

En langage C :

```
VAR     Int Age, Som, Nombre ;
        Float Prix, Moyenne ;
        Chat Chiffre ;
        Chat Chaine [20] ;
        Bool Gagner ;
```

### *Identificateur d'une constante ou variable*

Le nom d'une constante ou d'une variable est appelé un *identificateur*. Un bon identificateur de constante ou de variable doit satisfaire certains critères, essentiellement :

- Il doit être Alphanumérique commençant obligatoirement par une lettre et ne contenant pas de caractères spéciaux (>, <, +, -, etc.).
- Il doit être compréhensible (significatif) et facilement mémorable.
- Il doit être unique dans l'algorithme et différent des mots clés.

### *Conseil : Une constante de type caractère est mise entre apostrophes.*

Dans un algorithme :

- *E* est *une variable* mais *'E'* est *une constante* de type Caractère.
- *'12'* est *une constante* de type Réel, *12* est *une constante* de type Entier, et *12.5* est *une constante* de type Réel.

*Complément : Forme de données et mémoire centrale.*

Réellement, une variable ou une constante est un espace mémoire (de la mémoire centrale) qui va contenir des données soit fixe (constante), soit qui change au fur et à mesure que le programme avance dans son exécution (variable). Cependant, à un instant donné, une variable ne peut contenir *qu'une seule donnée (valeur)*.

# 3. Présentation d'une expression

Une expression présente un calcul *arithmétique* ou *booléenne*.

## 3.1. Expression arithmétique

Une expression arithmétique est donnée sous la forme *X OP Y*, où :

• *X* et *Y* sont des opérandes qui peuvent être des constantes ou des variables.

• *OP* est un opérateur arithmétique (+, -, *, DIV, MOD, /).

En plus de ces éléments, une expression arithmétique peut contenir des parenthèses.

*Exemple : Expressions arithmétiques*

Deux expressions arithmétiques sont données comme suit :

- 2 * X +Y.
- (-4 div 2) * 5 + (1 / 2) MOD 3.

## 3.2. Expression booléenne

Une expression logique contient des opérandes, et des opérateurs de comparaison ou des opérateurs booléens. Elle peut être une expression de relation ou une expression logique.

- *Expression de relation (comparaison) :* c'est une expression composée d'opérandes (qui peuvent être des variables ou des constantes) ayant des valeurs numériques, et des opérations de comparaison (>, <, ≥, ≤, =, ≠).
- *Expression logique :* c'est une expression qui contient des opérandes de type booléen et des opérateurs logiques (ET, OU, NON).

## 3.3. Les opérations permises sur les types de bases

Chaque type de base possède un ensemble d'opérations qui peuvent être appliquées aux données de ce type. Les opérations que nous pouvons utiliser dans un algorithme sont décrites dans le tableau

suivant :

| Types | Exemples | Opérations possibles | Symboles ou mots clés correspondants |
|---|---|---|---|
| Entier | -5, 15, 30 | Addition | + |
| | | Soustraction | - |
| | | Multiplication | * (pas x pour ne pas confondre avec la lettre x) |
| | | Division entière | DIV |
| | | Reste (modulo) | MOD |
| | | Comparaisons | <, ≤,>, ≥, =, ≠ |
| Réel | -15.5, 3.75, -2.62 | Addition | + |
| | | Soustraction | - |
| | | Multiplication | * |
| | | Division réelle | / (÷) |
| | | Reste (modulo) | <, ≤,>, ≥, =, ≠ |
| | | Comparaisons | <, ≤,>, ≥, =, ≠ |
| Caractère | 'a', 'B', '?' | Comparaisons | <, ≤,>, ≥, =, ≠ |
| Chaîne | 'abc', 'salem', '**1000**' | Concaténation | + (&) |
| Booléen | Vrai, Faux | Comparaisons | <, ≤,>, ≥, =, ≠ |
| | | Conjonction | ET |
| | | Disjonction | OU |
| | | Négation | NON |

*Tableau décrivant l'ensemble des opérations permises sur chaque type de base.*

## 3.4. Évaluation des expressions

L'ordre des opérations constituant une expression obéit aux règles suivantes :

- Les opérations entre parenthèses sont évaluées les premières.
- Les opérations sont évaluées *de gauche à droite* lorsqu'elles ont des priorités égales. Sinon, *de la plus prioritaire à la moins prioritaire* lorsqu'elles ont des priorités inégales.

Les priorités des opérateurs sont résumées dans le tableau suivant :

| Priorités | Opérateurs |
|---|---|
| 1 | NON, - (unitaire) |
| 2 | *, /, DIV, MOD, ET |
| 3 | +, - (binaire), OU |
| 4 | <, <=, =, >, <> |

*Tableau illustrant les priorités des opérateurs.*

# Les instructions de base d'un algorithme

X

Dans un algorithme, on distingue principalement deux sortes d'instructions : *les instructions de traitement (pour effectuer des opérations)*, et *les instructions de contrôle (pour indiquer la manière dont les opérations sont effectuées)*. Dans le cadre de ce chapitre, nous illustrons les instructions de traitement uniquement. Le deuxième type d'instructions sera décrit en détails dans les prochains chapitres.

Il existe trois instructions de traitements de base : *la lecture*, *l'écriture*, et *l'affectation*.

## 1. L'instruction LIRE (scanf)

Elle est utilisée pour attribuer une valeur à une variable. Dans le cas d'un programme, la valeur est lue à partir du clavier de l'ordinateur lors de l'exécution du programme. Ensuite, elle stockée dans la zone mémoire réservée pour de la variable.

### *Syntaxe : L'instruction LIRE (scanf)*

La syntaxe de l'instruction LIRE (READ) estp.39 :

```
1 LIRE (v1, v2, ..., vn) ;
```

En langage C :

```
1 scanf ("%d%d...%d", &v1, &v2, ..., &vn) ;
```

Où : v1, v2, ..., vn sont des identificateurs de variables de même entier.

Les *n* valeurs introduites au clavier sont affectées dans l'ordre aux variables v1, v2, ..., vn (la première à v1, la deuxième à v2, etc.).

Noté bien que dans le langage C, vous devez ajouter %d et & tel que illustré dans l'exemple ci-haut. Spécifiquement, %d signifie que scanf attend qu'un entier soit saisi par clavier. Par ailleurs, chaque type de donnée possède son spécificateur de format. Par exemple : %f pour le float, %lf pour le double, %c pour le char, etc.

### *Attention : Garder en tête*

- Il est impossible de lire la valeur d'une constante par l'instruction LIRE. Car, une constante une

fois déclarée, nous ne pouvons pas modifier sa valeur.

- Il est interdit de lire une expression arithmétique ou logique ; LIRE (a+b) est impossible.

### *Remarque : Fonctionnement d'affichage d'une valeur.*

Dans un programme, quand l'ordinateur rencontre une instruction de lecture :

1. Il s'arrête.
2. On tape une valeur au clavier.
3. Aussitôt que la touche "Entrée" est frappée, il prend cette valeur et la stocke dans la case mémoire de la variable.
4. Il continue l'exécution de programme.

## 2. L'instruction ÉCRIRE (printf)

Elle permet d'écrire (afficher) les résultats sur l'écran.

### *Syntaxe : L'instruction ÉCRIRE (WRITE)*

La syntaxe de l'instruction ÉCRIRE estp.39 :

```
ÉCRIRE (v1, v2, ..., vn) ;
```

Les valeurs des variables : v1, v2, ..., vn sont affichées à l'écran.

En langage C :

```
Nous prenons un exemple de l'affichage d'un entier a et d'un réel b.
printf("l'entier = %d et le réel = %f", a, b);
```

L'ordinateur cherche les valeurs de a et b dans la mémoire et affiche sur l'écran la phrase : l'entier = (valeur de a) et le réel = (valeur de b).

### *Remarque : Affichage avec l'instruction ÉCRIRE*

L'instruction ÉCRIRE permet aussi d'afficher des constantes (en général des textes) et des valeurs d'expressions. On peut par exemple avoir :

- Si nous avons a = 3 et b = 1, ÉCRIRE (a, 'plus', b, '=', a + b) ; affichera sur l'écran : *3 plus 1 = 4*.
- ÉCRIRE ('Bonjour, ceci est un texte à afficher') ; affichera sur l'écran le texte : *Bonjour, ceci est un texte à afficher*.

## 3. L'instruction d'affectation

Elle permet de donner une valeur déjà existante en mémoire centrale à une variable.

### *Syntaxe : L'opération d'affectations*

L'opération d'affectation se fait comme suitp.39 :

```
1 Nom_de_variable <- valeur ;
```

En langage C :

```
1 Nom_de_variable = valeur ;
```

Où *valeur* peut être une constante, une autre variable, ou une expression.

### *Complément*

Tout comme l'instruction LIRE. Nous ne pouvons pas affecter une valeur à une constante.

### *Exemple : Quelques opérations d'affectation*

Nombre <- 5 ;

Som <- Nombre ;

Moyenne <- (note1 + note2)/2 ;

### *Attention : Garder en tête*

- Dans l'instruction Som <- nombre, la valeur de la variable nombre est au préalablement stockée en mémoire, et ne change pas suite à l'opération de l'affectation.
- Réel <- Entier est autorisé.
- Entier <- Réel n'est pas autorisé.
- Chaîne de caractères <- Caractère est autorisé.
- Caractère <- Chaîne de caractères n'est pas autorisé.


## Exemple complet.

Écrire un algorithme qui calcule la moyenne d'un étudiant en informatique pour l'EMD1, en passant par l'étape d'analyse.

1. *Analyse du problème :*
   Avant de penser à une solution, il faut tout d'abord analyser le problème pour le but de sa compréhension. Comme expliquer en introduction, l'analyse est faite en répondant aux trois questions :
   - Les données (entrées) : Note_EMD et Note_Evaluation.
   - Les résultats (sorties) : La moyenne.
   - Les traitements à faire : (Note_EMD * 2 + Note_Evaluation)/3.
2. *Les trois blocs de l'algorithme :*
   - Lecture (Note_EMD), Lecture (Note_Evaluation) ;
   - Moyenne <- (Note_EMD * 2 + Note_Evaluation) / 3;
   - Écriture (Moyenne).

```
1 ALGORITHME moyenneEtudiant ;
2
3 VAR  Note_Evaluation, Note_EMD, Moyenne : Réel ;
4
5 DÉBUT
6
7   LIRE(Note_Evaluation) ; /* Les entrées */
8   LIRE(Note_EMD) ;
9
10  Moyenne <- (Note_EMD * 2 + Note_Evaluation)/3; /* Les traitements */
11
12  ÉCRIRE(Moyenne) ; /* Les sotries */
13
14 FIN.
```

# Exercice

XII

*[solution n°11 p.36]*

Quelle est l'écriture qui permet d'ajouter 5 à la variable "*prix*" ?

- `prix <- 5 ;`
- `prix <- prix * prix ;`
- `prix <- prix + 5 ;`

# La trace d'exécution d'un algorithme

XIII

Elle permet de suivre le déroulement (l'exécution manuelle) d'un l'algorithme pas à pas (instruction après instruction) afin de s'assurer de son bon fonctionnementp.39 . La trace d'exécution permet de :

- Vérifier les valeurs des variables après exécution de chaque instruction.
- S'assurer de la validité de l'algorithme. Essentiellement, s'assurer que l'algorithme se termine, et que son résultat est correct.

## Exemple de la trace d'un algorithme.

Que sera-t-il affiché à l'écran par l'algorithme suivant :

```
1  ALGORITHME Trace ;
2
3  VAR X, Y, Z : entier ;
4
5  DÉBUT
6  X  <- 3 ;
7  Y  <- 7 ;
8  Z  <- X ;
9  X <- Y ;
10 Z  <- Z + 1 ;
11 ÉCRIRE (2, X, Y) ;
12 ÉCRIRE  (X + 1 / 2) ;
13 FIN.
```

| Les variables | X | Y | Z | Écran |
|---|---|---|---|---|
| A la déclaration | ? | ? | ? | / |
| X ← 3 | 3 | ? | ? | / |
| Y ← 7 | 3 | 7 | ? | / |
| Z ← X | 3 | 7 | 3 | / |
| X ← Y | 7 | 7 | 3 | / |
| Z ← Z +1 | 7 | 7 | 4 | / |
| Écrire (2, X, Y) | 7 | 7 | 4 | 2 7 7 |
| Écrire (X + 1 / 2) | 7 | 7 | 4 | 7.5 |

*Tableau illustrant la trace d'éxecution de l'algorithme Trace.*

# Exercice

XV

*[solution n°13 p.36]*

# 0. Exercice : Choisissez la bonne réponse.

L'instruction qui me permet d'attribuer une valeur à la variable "nom" partir du clavier est :

- `LIRE (nom) ;`
- `ÉCRIRE (nom) ;`
- `nom <- 'Kamel' ;`

# 0. Exercice : Cochez la (les) bonne(s) réponse(s).

Un algorithme permet :

- ☐ De mieux comprendre le fonctionnement des logiciels.
- ☐ Par une suite finie d'instructions de résoudre un problème.
- ☐ De définir l'architecture d'un ordinateur.

# 0. Exercice : Cochez la (les) bonne(s) réponse(s).

Quelle est la différence entre une variable et une constante?

- ☐ Il n'y ‘a aucune différence entre une variable et une constante.
- ☐ La variable ne change jamais de nom par contre la constante change toujours.
- ☐ La valeur d'une variable varie durant le déroulement de l'algorithme.

# Test

XIX

Exercice : Choisir la bonne réponse.

*[solution n°14 p.37]*

Après l'exécution de l'algorithme suivant, quelle sera la valeur de "c" ? si a vaut 7 et si "b" vaut -5 ?

```
ALGORITHME Exemple ;
VAR
a <- 7 ;
b <- -5 ;
DÉBUT
c <- a - b ;
c <- c + a ;
FIN.
```

- 19
- 12
- -12

Exercice

*[solution n°15 p.37]*

Remplir le trou avec les mots adéquats.

Pour déclarer la constante "X=2", il faux écrire _____ juste après l'entête de l'algorithme.

Exercice

*[solution n°16 p.38]*

Caractère (Char) | Variable | Booléen (Boolean) | Affectation | LIRE (READ)

ÉCRIRE (WRITE) | Constante | Expression | Réel (Real) | Chaîne de caractères (String)

Entier (Integer)

| Types de données | Formes de données | Instructions |
|---|---|---|
| | | |

# Solutions des exercices

### > **Solution** n°1

Exercice p. 6

En algorithmique, toutes les données à traiter sont d'une même nature.

- ○ Vrai
- ◉ Faux

### > **Solution** n°2

Exercice p. 6

L'instruction qui permet de lire une donnée simple à partir du clavier est l'instruction SI ... SINON.

- ○ Vrai
- ◉ Faux

### > **Solution** n°3

Exercice p. 6

L'instruction LIRE est une instruction d'opération.

- ◉ Vrai
- ○ Faux

### > **Solution** n°4

Exercice p. 7

Cochez la (les) bonne(s) réponse(s) si elle(s) existe(nt).

- ☑ Un algorithme présente une solution automatique à un problème.
- ☐ Un algorithme est constitué de variables uniquement.
- ☐ Un algorithme peut s’exécuter sur un ordinateur.

- ☐ On peut réaliser un algorithme sans comprendre le problème à résoudre.
- ☑ Un algorithme contient souvent plusieurs type d'instructions.
- ☑ On ne peut pas faire de la programmation sans se disposer d'un ordinateur.

## > **Solution** n°5

Exercice p. 8

L'unité central est un composant software.

- ○ Vrai
- ◉ Faux

## > **Solution** n°6

Exercice p. 8

Le clavier d'un ordinateur :

- ☑ Est un composant hardware d'un ordinateur.
- ☐ Est un composant software d'un ordinateur
- ☑ Fait partie des unités d'entrée/sortie d'un ordinateur.
- ☐ Fait partie des unités de traitement d'un ordinateur.

## > **Solution** n°7

Exercice p. 8

Définissez brièvement un système d'exploitation.

C'est un ensemble de programmes qui dirige l'utilisation des capacités d'un ordinateur par des logiciels applicatifs.

## > **Solution** n°8

Exercice p. 9

| Types de données | Formes de données | Instructions |
|---|---|---|
| Entier (Integer) | Constante | LIRE (READ) |
| Réel (Real) | Variable | ÉCRIRE (WRITE) |
| Caractère (Char) | Expression | Affectation |
| Booléen (Boolean) | | |
| Chaîne de caractères (String) | | |

## > **Solution** n°9

Exercice p. 14

Un algorithme se construit :

- ○ Après l'écriture d'un programme.
- ⊙ Avant l'écriture d'un programme.
- ○ L'ordre de construction n'a pas d'importance.

## > **Solution** n°10

Exercice p. 25

Quelle est l'écriture qui permet d'ajouter 5 à la variable "*prix*" ?

- ○ `prix <- 5 ;`
- ○ `prix <- prix * prix ;`
- ⊙ `prix <- prix + 5 ;`

## > **Solution** n°11

Exercice p. 28

Exercice : Choisissez la bonne réponse.

L'instruction qui me permet d'attribuer une valeur à la variable "nom" partir du clavier est :

- ⊙ `LIRE (nom) ;`

- ÉCRIRE (nom) ;
- nom <- 'Kamel' ;

Exercice : Cochez la (les) bonne(s) réponse(s).

Un algorithme permet :

- ☐ De mieux comprendre le fonctionnement des logiciels.
- ☑ Par une suite finie d'instructions de résoudre un problème.
- ☐ De définir l'architecture d'un ordinateur.

Exercice : Cochez la (les) bonne(s) réponse(s).

Quelle est la différence entre une variable et une constante?

- ☐ Il n'y ‘a aucune différence entre une variable et une constante.
- ☐ La variable ne change jamais de nom par contre la constante change toujours.
- ☑ La valeur d'une variable varie durant le déroulement de l'algorithme.

## > **Solution** n°12

Exercice p. 32

Après l'exécution de l'algorithme suivant, quelle sera la valeur de "c" ? si a vaut 7 et si "b" vaut -5 ?

```
ALGORITHME Exemple ;
VAR
a <- 7 ;
b <- -5 ;
DÉBUT
c <- a - b ;
c <- c + a ;
FIN.
```

- ◉ 19
- ○ 12
- ○ -12

## > **Solution** n°13

Exercice p. 32

Remplir le trou avec les mots adéquats.

Pour déclarer la constante "X=2", il faux écrire `CONST X=2;` juste après l'entête de l'algorithme.

## > **Solution** n°14

Exercice p. 33

| Types de données | Formes de données | Instructions |
|---|---|---|
| Entier (Integer) | Constante | LIRE (READ) |
| Réel (Real) | Variable | ÉCRIRE (WRITE) |
| Caractère (Char) | Expression | Affectation |
| Booléen (Boolean) | | |
| Chaîne de caractères (String) | | |

# Références

*1*
L. Baba-Hamed, S. Hocine. Algorithmique et structures de données : Cours et exercices avec solutions. Office des Publications Universitaires, 2006

*2*
M. C. Belaid. Algorithmique et programmation en Pascal, cours, exercices et travaux pratique avec corrigés. Edition Les Pages Bleues, 2004.

*3*
M. C. Belaid. Algorithme et programmation en Pascal. Edition les pages bleus, 2006.

*4*
T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. Algorithmique : Cours avec 957 exercices et 158 problèmes. Édition DUNOD, 3ème édition, 2010.

*5*
J. Courtin. Initiation à l'algorithmique et aux structures de données. Edition DUNOD, 1998.

*6*
M. Divay. Algorithmes et structures de données génériques. Edition Dunod, 2004.

*7* L. Goldschlager and A. Lister. Informatique et algorithmique. InterEditions. 1986.

*8*
https://9alami.info/cours-informatique/liste-des-modules/lecon1-notion-dalgorithme (Consulté le 09/03 /2021)

*9*
https://www.youtube.com/playlist?list=PL2aehqZh72Lumvy4tSekr6Rzcgwn15MLI (Consulté le 05/04 /2021)