

# Support de cours – Algorithmique (Partie 1 & Partie 2)

## Objectifs généraux du module Algorithmique

À la fin de ce module, l'apprenant sera capable de : - Comprendre la notion d'algorithme et la logique de programmation. - Décomposer un problème en étapes logiques et structurées. - Manipuler les concepts : variables, conditions, boucles et fonctions. - Manipuler des tableaux simples et tableaux d'objets (préparation au JSON / JavaScript). - Concevoir un mini-projet algorithmique complet (proche d'un futur projet JavaScript).

---

## PARTIE 1 : INTRODUCTION À L'ALGORITHMIQUE

### 1. Qu'est-ce qu'un algorithme ?

Un **algorithme** est une suite d'instructions permettant de résoudre un problème donné.

Exemples d'algorithmes dans la vie quotidienne : - Préparer un café ☕ - Envoyer un email - Retirer de l'argent d'un distributeur automatique

Un programme informatique n'est qu'une traduction d'un algorithme dans un langage.

---

### 2. Notion de variable

Une **variable** est un espace dans la mémoire permettant de stocker une information.

Pseudo-code :

```
Début
  variable nom ← "Zakaria"
  variable age ← 27
Fin
```

### 3. Les conditions (SI / ALORS / SINON)

Elles permettent de prendre une décision.

Pseudo-code :

```
Si age >= 18 alors
  Afficher "Majeur"
Sinon
  Afficher "Mineur"
FinSi
```

---

## 4. Les boucles (Pour / Tant que)

Elles permettent de répéter une instruction plusieurs fois.

### Boucle POUR (connue à l'avance)

```
Pour i de 1 à 5
  Afficher i
FinPour
```

### Boucle TANT QUE (conditionnelle)

```
Tant que i < 5
  i ← i + 1
FinTantQue
```

---

## 5. Les fonctions (modularisation)

Les **fonctions** permettent de réutiliser un bloc d'instructions.

```
Fonction addition(a, b)
  retourner a + b
FinFonction
```

---

## 6. Exercices (progressifs)

Exercice	Objectif
1. Afficher la somme de deux nombres	Variables + calcul
2. Tester si un nombre est pair ou impair	Condition
3. Afficher les nombres de 1 à 10	Boucle

Exercice	Objectif
4. Trouver la plus grande note dans un tableau	Tableau + boucle
5. Créer une fonction <code>moyenne(a, b)</code>	Fonction

## MINI-PROJET ALGORITHMIQUE (Partie 1)

Calculer la moyenne de deux notes et afficher "Admis" ou "Refusé".

Pseudo-code :

```
Début
  Lire note1
  Lire note2
  moyenne ← (note1 + note2) / 2

  Si moyenne >= 10 alors
    Afficher "Admis"
  Sinon
    Afficher "Refusé"
  FinSi
Fin
```

## PARTIE 2 : CONCEPTS AVANCÉS

(Préparation directe au JavaScript et à la manipulation des données)

### 1. Tableaux et tableaux d'objets

#### Tableau simple

```
notes ← [12, 16, 8, 17, 14]
```

#### Tableau d'objets (prépare au JSON et API)

```
etudiants ← [
  { nom: "Sara", note: 17 },
  { nom: "Yassine", note: 12 },
  { nom: "Amine", note: 8 }
]
```

## 2. Recherche (dans une liste ou un tableau d'objets)

Pseudo-code : rechercher si la note 16 existe.

```
trouver ← FAUX
Pour chaque element dans notes
  Si element = 16 alors
    trouver ← VRAI
  FinSi
FinPour

Afficher trouver
```

## 3. Tri (croissant / décroissant)

Pseudo-code (méthode simple) :

```
Pour i de 1 à taille - 1
  Pour j de i+1 à taille
    Si tableau[i] > tableau[j] alors
      échanger tableau[i] et tableau[j]
    FinSi
  FinPour
FinPour
```

## 4. Fonctions avancées (modularisation)

Exemple de découpage d'un programme :

```
Fonction saisirNotes()
Fonction calculerMoyenne()
Fonction afficherResultat()
```

Le programme principal appelle simplement les fonctions.



## MINI-PROJET ALGORITHMIQUE AVANCÉ

Gestion d'une liste d'étudiants (nom, âge, note) avec tri et recherche

Pseudo-code :

Début

  Lire nombreEtudiants

  Pour i de 1 à nombreEtudiants

    Lire nom

    Lire age

    Lire note

    Ajouter {nom, age, note} à etudiants

  FinPour

  Trier etudiants par note décroissante

  Afficher "Le meilleur étudiant est : ", etudiants[0].nom

Fin

---

## Résultat attendu

Compétence	Validé
Variables, conditions, boucles	
Tableaux et objets (version JSON)	
Tri, recherche, filtrage	
Structuration (fonctions, modularisation)	
Prêt pour JavaScript (DOM + API + projet final)	

---

## Conclusion

Une bonne maîtrise de l'algorithmique facilite l'apprentissage de **JavaScript**, puis de toute autre technologie (frameworks, API, etc.). C'est la base de la logique développeur.