



## Licence 1 – Pure Developer (2022 - 2023)

### Algorithmes et structures de données 1

Dr. Pegdwendé Nicolas Sawadogo

[nicolas.sawadogo@uv.bf](mailto:nicolas.sawadogo@uv.bf)  
<http://pegdwende.net>

Février 2023



Introduction

### Déroulement du cours

- Cours théoriques ≈ 30h
  - ▶ 5 séances live de 4h
  - ▶ 10h de travail complémentaire
- TD ⇒ 18h
  - ▶ 5 séances de 3h
  - ▶ 1 séance d'évaluation
- Evaluation
  - ▶ 1 évaluation sous forme de QCM (50% de la note finale)
  - ▶ 1 évaluation sous forme de TD à rendre (50 %)

### Plan

- 1 Introduction
- 2 Éléments de base d'un algorithme
- 3 Structures de contrôle
- 4 Types de données avancés
- 5 Fonctions et procédures



Pegdwendé N. Sawadogo

Algorithmes et Structures de données 1

Février 2023

2 / 60

Introduction

### Objectifs

Ce cours d'algorithmique vise à vous apprendre à programmer :

- ▶ Résoudre un problème en plusieurs étapes
- ▶ Comprendre le fonctionnement d'un programme sur l'ordinateur
- ▶ "Parler" à l'ordinateur
- ▶ Optimiser la gestion des ressources (mémoire, processeur)
- ▶ Ecrire un programme compréhensible par d'autres
- ▶ Comprendre un programme écrit par d'autres



# Notion d'algorithme

*Qu'est-ce qu'un algorithme?*



- **Muhammad ibn Mūsā al-Khwārizmī** est un mathématicien persan qui a vécu aux 8<sup>e</sup> et 9<sup>e</sup> siècles.
- Le mot *algorithme* est issu d'une déformation latinisée de son nom
- Al-Kwarizmi n'a pas inventé l'algorithme, mais a réalisé une classification des algorithmes connus à son époque

# Notion d'algorithme

*Qu'est-ce qu'un algorithme?*

## Définition

Un algorithme est une suite finie et non ambiguë d'instructions et d'opérations permettant de résoudre un problème.  
Le domaine qui étudie les algorithmes est appelé l'**algorithmique**

## Exemples

- ▶ Calcul d'une somme/moyenne/produit, etc.
- ▶ Réalisation d'une recette de cuisine
- ▶ Résolution d'un casse tête (Rubik's cube par exemple)
- ▶ Production d'un tissu
- ▶ Etc.

# Notion d'algorithme

*Lien entre ordinateurs et algorithmes*

## Exercice n°1

Qu'est-ce qu'un ordinateur?

# Notion d'algorithme

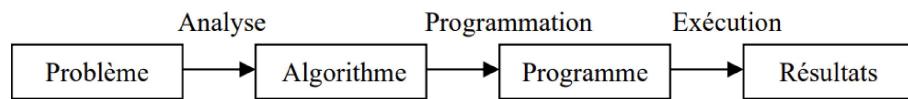
*Lien entre ordinateurs et algorithmes*

- Les algorithmes existaient déjà bien avant l'invention du premier ordinateur. Ils sont historiquement utilisés pour la réalisation d'opération arithmétiques
- Les algorithmes peuvent être exécutées par des ordinateurs, qui (à l'inverse de l'humain) ne font pas d'erreur et sont insensibles aux tâches répétitives.
- L'ordinateur est donc le support idéal d'exécution des algorithmes, mais pas le seul.
- Plusieurs domaines d'application des algorithmes sur ordi :
  - ▶ Cryptographie
  - ▶ Routage de l'information
  - ▶ Traitement d'images
  - ▶ Intelligence artificielle

# Notion d'algorithme

Lien entre algorithmes et programmation

- Tout problème à programmer doit être résolu, d'abord sous forme d'algorithme, puis converti en programme dans le langage de votre choix.
- Un algorithme est indépendant du langage de programmation utilisé
- Le programme est une traduction d'un algorithme en langage machine, compréhensible par l'ordinateur (Ex : langage C).



# Rappel du sommaire

## 1 Introduction

- 2 Éléments de base d'un algorithme
  - Syntaxe algorithmique
  - Types de base
  - Variables et Opérateurs
  - Syntaxe algorithmique 2

## 3 Structures de contrôle

## 4 Types de données avancés

## 5 Fonctions et procédures

# Syntaxe algorithmique

- Il existe des conventions de syntaxes et de formalismes pour l'écriture d'un algo.
- Plusieurs variantes de langage algorithmique existent :
  - ▶ Selon la langue
  - ▶ Selon le formalisme choisi : nous utiliserons le formalisme EXALGO<sup>1</sup>
- EXALGO est un mini langage permettant de fixer des règles permettant d'écrire des algorithmes
- EXALGO des chaînes de caractères alphanumériques, des signes opératoires, des mot-clés réservés, et de signes de ponctuation , règles de ponctuation et syntaxiques à utiliser.

# Syntaxe algorithmique

## Squelette d'un algorithme

---

### Algorithme 1: NomAlgorithm

**Entrées:** // (Liste des entrées)

**Résultat:** // (Résultat obtenu après l'exécution de l'algorithme)

**Variables:** // (Variables intermédiaires utilisées)

**Début**

| // (Instructions)

**Fin**

- 
- ▶ Un algorithme commence par le mot clé *Algorithme* suivi par le nom de l'algorithme (le numéro est facultatif)
  - ▶ Le nom de l'algorithme doit être évocateur
  - ▶ Il faut éviter l'inclusion d'espaces dans le nom d'un algorithme
  - ▶ Il faut avoir une écriture soignée et respecter l'indentation

# Syntaxe algorithmique

## Exemple

(Exemple)

### Algorithme 2: NomAlgorithme

```
Entrées: // (Liste des entrées)
Résultat: // (Résultat obtenu après l'exécution de l'algorithme)
Variables: // (Variables intermédiaires utilisées)
Début
| Afficher("Bonjour à vous ! ");
Fin
```

- ▶ L'algorithme 2 affiche à l'écran le texte "Bonjour à vous!"
- ▶ Le mot clé **Afficher()** sert à donner à l'ordinateur d'afficher quelque chose à l'écran

## Notion de type abstrait

### Définition

Un **type abstrait** est un triplet  $\langle N, \mathcal{V}, \mathcal{O} \rangle$  avec :

- ▶  $N$  : un nom;
- ▶  $\mathcal{V}$  : un ensemble de valeurs;
- ▶  $\mathcal{O}$  : un ensemble d'opérations applicables.

Ces types abstraits sont associés aux données qui peuvent être manipulées dans un algorithme.

Les types abstraits de base en algorithmique sont :

- ▶ entier;
- ▶ réels;
- ▶ caractères;
- ▶ booléens.

# Syntaxe algorithmique

## Exercice

### Exercice n°2

Ecrire un algorithme qui affiche à l'écran "Je sais programmer"

## Description des types abstraits de base

### Type entier

- ▶  $\mathcal{V}$  : ensemble des nombres relatifs (ensemble  $\mathbb{Z}$  en maths);
- ▶  $\mathcal{O}$  : addition, soustraction, multiplication, division, division entière, modulo, etc.

### Type réel

- ▶  $\mathcal{V}$  : ensemble des nombres réels (ensemble  $\mathbb{R}$  en maths);
- ▶  $\mathcal{O}$  : addition, soustraction, multiplication, division.

### Type caractère ("car" en écriture algorithmique)

- ▶  $\mathcal{V}$  : ensemble des caractères alphanumériques et spéciaux : 'a', 'b', 'c', '9', '!', etc. Elles sont notées entre cotes simples ('')
- ▶  $\mathcal{O}$  : addition, soustraction.

### Type booléen ("bool" en écriture algorithmique)

- ▶  $\mathcal{V}$  : Vrai, Faux
- ▶  $\mathcal{O}$  : Opérateurs logiques (ET, OU, NON, etc.)

# Notion de variables

## Définition

### Définition

Une **variable** est un triplet  $\langle N, T, V \rangle$  avec :

- ▶  $N$  : un nom;
- ▶  $T$  : un type abstrait auquel la variable est associée;
- ▶  $V$  : la valeur stockée dans la variable.

Chaque variable représente un emplacement de la mémoire de l'ordinateur. En créant une variable, on alloue de l'espace mémoire correspondant au type.

- Le nom d'une variable doit contenir uniquement des caractères alphanumériques.
- La valeur d'une variable peut changer pendant l'exécution d'un algorithme

# Manipulation de variables

## Opérateur d'affectation

- Cet opérateur permet d'attribuer une valeur à une variable. C'est cette valeur qui est stockée dans la mémoire de l'ordinateur
  - ▶ Syntaxe : nomVariable ← valeur;
  - ▶ Exemples :
 

```
age ← 18;
moyenne ← 12.5;
estMajeur ← vrai;
```
- L'affectation écrase l'ancienne valeur si elle existe
- Une variable peut recevoir par l'affectation ;
  - ▶ Une simple valeur :
 

```
anneeNaissance ← 2004
anneeActuelle ← 2023
```
  - ▶ Le résultat d'une opération mathématique :
 

```
age ← 2023 - 2004;
```
  - ▶ Le résultat d'une opération sur une variable :
 

```
age ← anneeActuelle - anneeNaissance;
```

# Manipulation de variables

## Définition de variables

- Définition d'une variable unique :

- ▶ Syntaxe : **var** nomVariable : Type
- ▶ Exemples :
 

```
var age : entier;
var moyenne : réel;
var estMajeur : bool;
```

- Définition de plusieurs variables :

- ▶ Syntaxe :
 

```
var nomVariable_1 : Type_1,
nomVariable_2 : Type_2,
...
nomVariable_n : Type_n;
```
- ▶ Exemple : **var** age\_1 : entier,  
moyenne\_2 : réel,  
estMajeur\_n : bool;

# Manipulation de variables

## Exercice

### Exercice n°3

En exécutant la suite d'instructions suivante, quelle valeur fera t-on s'afficher à l'écran ?

```
var note : réel ;
note ← 15.0 ;
note ← note + 2.0 ;
Afficher( note ) ;
```

# Manipulation de variables

Exercice

## Exercice n°4

En exécutant la suite d'instructions suivante, quelles valeurs fera-t-on s'afficher à l'écran ?

```
var panierBleu : entier ,
    panierRouge : entier ,
    panierVert : entier ;
panierBleu ← 1 ;
panierRouge ← 2 ;
panierVert ← 3 ;
panierBleu ← 5 ;
panierVert ← panierBleu + panierRouge ;
panierRouge ← 1 ;
Afficher( panierBleu ) ;
Afficher( panierVert ) ;
Afficher( panierRouge ) ;
```

# Syntaxe algorithmique 2

Squelette d'un algorithme

---

### Algorithme 1: NomAlgorithme

**Entrées:** // (Liste des entrées)

**Résultat:** // (Résultat obtenu après l'exécution de l'algorithme)

**Variables:** // (Variables intermédiaires utilisées)

**Début**

| // (Instructions)

**Fin**

---

- ▶ Les **entrées** représentent les informations à fournir par l'utilisateur pour l'exécution de l'algorithme.
- ▶ Le **résultat** représente ce qu'on obtient après l'exécution de l'algorithme.
- ▶ Les **variables** intermédiaires servent à manipuler des informations à l'intérieur de l'algorithme.

# Syntaxe algorithmique 2

Exercice

## Exercice n°4

Ecrire un algorithme qui récupère la date de naissance d'une personne et affiche son âge

# Instructions Afficher et Récupérer

Instruction Afficher

- ▶ La fonction **Afficher()** ou **Ecrire()** permet au programme d'afficher des messages (à l'écran par exemple) afin que l'utilisateur puisse les lire.
- ▶ Elle peut prendre en paramètre une valeur ou une variable

```
var nombreGarcons : entier ;
var nombreFilles : entier ;
var nombreEnfants : entier ;
Afficher( "Combien de garçons avez-vous ?" ) ;
Récuperer( nombreGarcons ) ;
Afficher( "Combien de filles avez-vous ?" ) ;
Récuperer( nombreFilles ) ;
nombreEnfants ← nombreGarcons + nombreFilles ;
Afficher( "Vous avez" , nombreEnfants , "enfants" ) ;
```

# Instructions Afficher et Récupérer

## Instruction Récupérer

- ▶ La fonction **Récuperer()** ou **Lire()** permet d'accueillir des valeurs saisies par l'utilisateur
- ▶ Elle prend en paramètre une ou plusieurs variables dont les valeurs seront entrées (au clavier).

```
var nombreGarcons : entier ;
var nombreFilles : entier ;
var nombreEnfants : entier ;
Afficher( "Combien de garçons avez-vous ?" ) ;
Récuperer( nombreGarcons ) ;
Afficher( "Combien de filles avez-vous ?" ) ;
Récuperer( nombreFilles ) ;
nombreEnfants ← nombreGarcons + nombreFilles ;
Afficher( "Vous avez" , nombreEnfants , "enfants" ) ;
```

## Rappel du sommaire

1 Introduction

2 Éléments de base d'un algorithme

3 Structures de contrôle

- Structures conditionnelles
- Boucles

4 Types de données avancés

5 Fonctions et procédures

# Instructions Afficher et Récupérer

## Exercice

### Exercice n°5

Ecrire un algorithme qui récupère le prix HT d'un produit ainsi que le taux de TVA à appliquer puis afficher le montant de la TVA et le prix TTC.

## Structures de contrôle

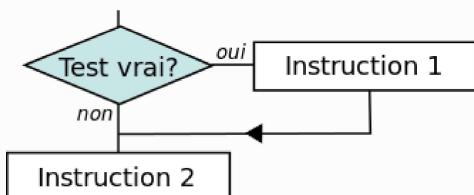
- Une structure de contrôle est une commande particulière permettant de conditionner ou de répéter l'exécution d'un bloc d'instruction.
- On distingue essentiellement deux types de structures de contrôle :
  - ▶ Les structures conditionnelles
  - ▶ Les boucles

# Condition Si

Si...FinSi

```
Si (condition) Alors
| //Instruction1;
FinSi
//Instruction2;
```

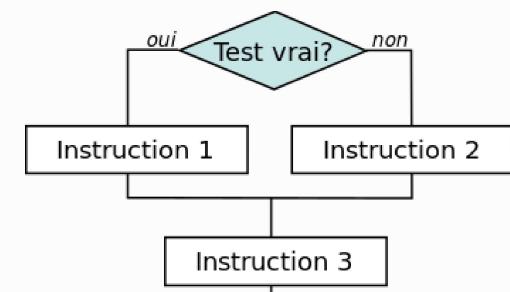
- ▶ Cette structure de contrôle permet de conditionner l'exécution d'un bloc d'instructions selon le résultat d'un test logique.
- ▶ L'Instruction1 s'exécute uniquement si la condition est respectée.



# Condition Si

Si...Sinon...FinSi

```
Si (condition) Alors
| //Instruction1;
Sinon
| //Instruction2;
FinSi
//Instruction3;
```



- ▶ Cette variante permet de conditionner l'exécution de deux blocs d'instructions alternatives.
- ▶ L'Instruction1 s'exécute si la condition est respectée, et l'Instruction2 si la condition n'est pas respectée.

# Condition Si

Exemples

```
var age :entier ,
    prixFormation :réel;
prixFormation ← 10000;
Afficher("Quel est votre age?");
Récupérer(age);
Si (age < 18) Alors
| prixFormation ← prixFormation - prixFormation * 0.25;
FinSi
Afficher("Le prix de la formation est de ", prixFormation, " F. CFA");
```

# Condition Si

Exercice

## Exercice n°6

Ecrire un algorithme qui récupère la nationalité d'un étudiant et détermine le montant de ses frais d'inscription à l'université suivant la règle suivante :

- Si l'étudiant est Burkinabè, les frais s'élèvent à 50000 F.
- Si l'étudiant est d'une autre nationalité, les frais s'élèvent à 250000 F.



# Boucle Pour

Exercice

## Exercice n°7

Ecrire un algorithme qui affiche la liste des 10 entiers suivant un nombre entré par l'utilisateur

## Exercice n°8

Ecrire un algorithme qui affiche la table de multiplication d'un nombre entré par l'utilisateur

## Exercice n°9

Ecrire un algorithme qui affiche les tables de multiplication de tous les nombres de 1 à 10

# Notion de tableau

## Définition

Un tableau est une liste ordonnée de  $N$  valeurs du même type.

- ▶ On appelle  $N$  la taille du tableau, et les valeurs qu'ils contient sont ses éléments.
- ▶ Chaque élément est repéré dans le tableau par son indice, un nombre entier compris entre 1 et  $N$

Indice	→	1	2	3	4	5	6	7	8	9
Élément	→	6	71	-8	12	93	5	-10	-34	7

# Rappel du sommaire

1 Introduction

2 Éléments de base d'un algorithme

3 Structures de contrôle

4 Types de données avancés

- Tableaux à 1 entrée
- Chaînes de caractères
- Tableaux à 2 entrées
- Structures

5 Fonctions et procédures

# Les tableaux en langage algorithmique

## Déclaration d'un tableau

- ▶ Tableau de 10 *notes* :
- var Notes : Tableau[10] entier;**
- ▶ Tableau de 5 *moyennes* :
- var Moyennes : Tableau[10] réel;**

## Définition / modification des valeurs d'un tableau

```
Moyennes[1] ← 18.5
Moyennes[2] ← 11.0
Moyennes[3] ← 15.5
Moyennes[4] ← 12.0
Moyennes[5] ← 15.0
```

## Accès aux valeurs d'un tableau

**Afficher(Moyennes[1])**

**Afficher(Moyennes[2] + Moyennes[3]))**





# Manipulation d'une structure

## Création d'un type structuré

### Type Etudiant

nom : chaîne  
prenom : chaîne  
age : entier

### FinType

## Création d'une variable structure

```
var monEtudiant :Etudiant;
```

## Accès aux valeurs d'un type structuré

```
monEtudiant.nom ← "SAWADOGO";  
monEtudiant.prenom ← "Pegdwende";  
monEtudiant.age ← 18;
```

## Rappel du sommaire

1 Introduction

2 Éléments de base d'un algorithme

3 Structures de contrôle

4 Types de données avancés

5 Fonctions et procédures

- Fonctions
- Procédures

# Exercice

## Exercice n°15

Créer une structure MaDate dans laquelle vous mettrez les informations sur la date d'aujourd'hui (jour du mois, mois, jour de la semaine, année, férié ou pas)

## Notion de fonction

### Définition

Une fonction est un algorithme qui peut être utilisé dans d'autres algorithmes.

- ▶ La fonction *peut* récupérer des informations en entrée appelées paramètres
- ▶ Elle *peut* retourner une seule information à la fin de son exécution avec l'instruction **RETOURNER**
- Les fonctions servent à la modularité du code
- Cela facilite le débogage et évite les répétitions
- Elles permettent aussi une meilleure lisibilité

# Ecriture d'une fonction

## Syntaxe

**Fonction** *Exemple(Liste de paramètres)*

**Variables:**

**Résultat:** valeur: entier

/\* Traitements (corps de la fonction)...

**Retourner** valeur;

**FinFonction**

- Une fonction peut contenir des variables locales, qui ne sont pas visibles à l'extérieur de la fonction
- Au cas où la fonction ne retourne pas de résultat, on retourne une valeur vide

# Ecriture d'une fonction

## Exemples

**Fonction** *SommeDe2Nombres(nombre1:entier , nombre2:entier)*

**Variables:**

**Résultat:** somme: entier

somme ← nombre1 + nombre2 ;

**Retourner** somme

**FinFonction**

**Fonction** *Age(anneeNaissance:entier)*

**Variables:**

**Résultat:** age: entier

age ← 2023 - anneeNaissance ;

**Retourner** age

**FinFonction**

# Appel d'une fonction

## Fonctions usuelles

- Il existe des fonctions fournies par les plateformes de programmation
- Ces fonctions sont communément organisées en modules
- Des exemples de fonctions prêtes-à-l'emploi :
  - **Afficher(valeur)** qui affiche à l'écran;
  - **Récupérer(variable)** lit une valeur saisie au clavier;
  - etc.

## Emploi

- Une fonction s'utilise en utilisant son nom, avec entre parenthèse la liste des instances de paramètres.  
Exemple : *Afficher("Bonjour")*;
- Une fonction peut être employée à l'intérieur d'une expression.  
*moyenne ← Somme(note1, note2) / 2;*

# Trace d'exécution d'une fonction

## Définition

La trace est un "compte-rendu" de l'exécution de l'algorithme avec des paramètres donnés. Elle est constituée en prenant une "photo" de toutes les variables de cet algorithme aux instants suivants :

- Au début;
- A chaque instruction intermédiaire
- A la fin
- La trace est représentée sous la forme d'un tableau
- Les colonnes sont les suivantes :
  - une première colonne représente le numéro de ligne de l'algorithme;
  - une colonne pour chaque variable,
  - une colonne de commentaire.
- Chaque ligne correspond à une instruction (hors-mis les lignes correspondant au début et à la fin).

## Trace d'exécution d'une fonction

Exemple : Exécuter "MaFonction(5, 5, 10);

**Fonction** *MaFonction(x:entier, a:entier, b:entier )*

Variables: tmp:entier

Résultat: y: entier

1      tmp  $\leftarrow$  a ;

2      a  $\leftarrow$  b ;

3      b  $\leftarrow$  tmp ;

4      y  $\leftarrow$  a \* x + b ;

5      Retourner y

FinFonction

N° ligne	x	a	b	y	tmp	commentaire
Début	5	5	10	-	-	Initialisation
ligne 1	5	5	10	-	5	La valeur de tmp change
ligne 2	5	10	10	-	5	La valeur de a change
ligne 3	5	10	5	-	5	La valeur de b change
ligne 4	5	5	10	55	5	La valeur de y change
Fin	5	5	10	55	5	La valeur de tmp change

## Récursivité

### Définition

Une fonction est dite récursive lorsqu'elle fait appel à elle-même

### Exemple :

**Fonction** *Factorielle(x:entier)*

Variables:

Résultat: y: entier

1      Si  $x < 1$  Alors

  |    y  $\leftarrow$  1 ;

2      Sinon

  |    y  $\leftarrow$  x \* Factorielle(x - 1) ;

3      FinSi

4      Retourner y

FinFonction

## Notion de procédure

### Définition

Une procédure est une fonction particulière qui ne retourne rien.

Dans ce cas, on peut :

- Soit omettre l'instruction **Retourner**;
- Soit retourner comme valeur "vide".

### Exemple :

**Procédure** *Bonjour(nom:chaine, sexe:booléen)*

Variables:

Résultat:

1      Si sexe = "vrai" Alors

  |    Afficher("Bonjour Monsieur ", nom );

2      Sinon

  |    Afficher("Bonjour Madame ", nom );

3      FinSi

4      FinProcedure

## Exercices