

---

# **Cours** **D'Algorithmique**

*Partie 1*  
*Opérations de Base et*  
*Structures de contrôle*

# **UE MIF113 : INFORMATIQUE**

## **EC: MIF 1131**

### **Cours:**

### **Algorithmiques et**

### **Structure des données**



# Présentation générale

---

## Unité d'Enseignement

**Titre : UE 113 INFORMATIQUE**

## Élément constitutif

**Titre : Algorithmique**

## Déroulé

Cours CM en ppt

TD Corrigés

TP notés

## Evaluation

**Travaux personnels / Travaux Dirigés 20 %**

**Contrôle des connaissances : 30% sous forme de QCM**

**Examen écrit : 50%**

**Examen de Rattrapage : 100%**

## Professeur responsable

Dr Moussa Dethie SARR

[mdsarr@univ-thies.sn](mailto:mdsarr@univ-thies.sn)

Numéro whatsapp : 775571817



# Algorithmique

---

- **Objectifs**

- Savoir écrire un algorithme à partir d'un problème donné



# Algorithmique

---

## •Plan

- I. : Présentation générale
- II. : Opérations de bases
- III. : Structures de contrôle
- IV. : Tableaux

# Algorithmique et Structure de données

## *Partie 1* *Présentation Générale*

# I - Présentation générale

---

## •Programmer pourquoi ? comment ?

### - Problématique

→ Informations **trop nombreuses, trop complexes** pour être traitées manuellement, **calculs répétitifs ou fastidieux**

→ Volonté d'automatiser certaines tâches

### - Intérêt

→ Le système informatique aide à la manipulation des informations, il permet de **collecter**, de **stocker** ces informations, de les **consulter**, de les **modifier**, d'y **effectuer des traitements** (tris, calculs) à volonté

### - Fonctionnement

→ Le système informatique reconnaît et sait exécuter un ensemble d'opérations élémentaires, appelées **instructions**

→ Un programme spécifiera l'ensemble des instructions à exécuter pour réaliser le traitement souhaité et leur enchaînement

# I - Présentation générale

---

## •Elaboration d'un programme

### - Formaliser le problème (scientifique ou de gestion)

→ Comprendre le problème (*lire attentivement l'énoncé !*)

→ Mettre en évidence les **informations nécessaires en entrée** (les données lues, les hypothèses) et **en sortie** (les données à écrire, les résultats attendus)

### - Formaliser le traitement

→ **Déterminer les calculs** que l'on va réaliser en recherchant la manière la plus simple et la plus systématique de résoudre le problème

→ **Déterminer les calculs intermédiaires** à réaliser et définir les données intermédiaires dont on aura besoin

→ **Réaliser un jeu d'essai** permettant de tester les différents cas, en particulier les cas limites et vérifier que les résultats obtenus sont ceux attendus

### - Ecrire l'algorithme correspondant

# From scratch ...

---

**Résoudre un problème**

**Indiquer où vous habitez**



Décrire le cheminement permettant d'arriver à la solution d'un problème décrit par un énoncé

Décrire le cheminement à emprunter pour arriver à votre habitation telle que décrit par votre adresse

# I - Présentation générale

---

**Résoudre un problème**

**Indiquer où vous habitez**



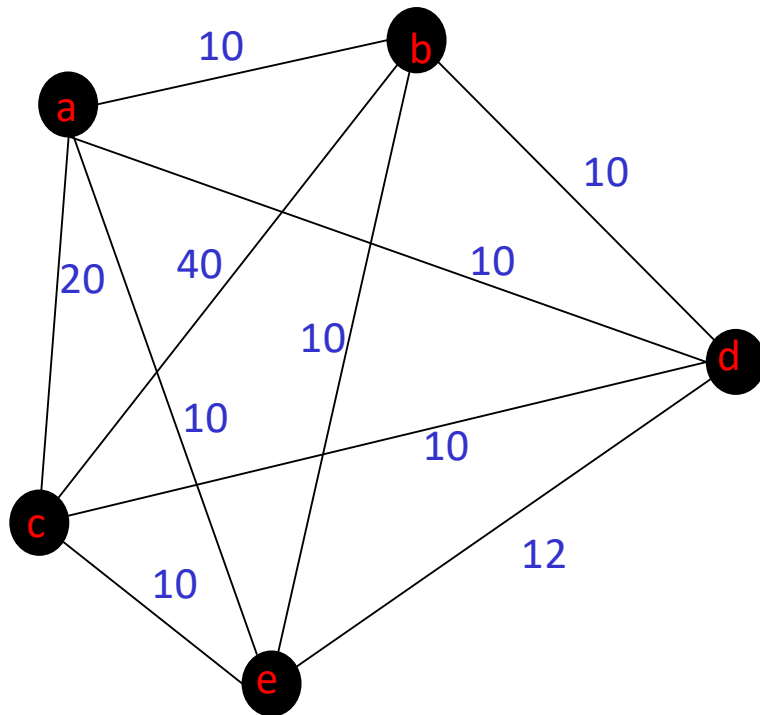
Décrire le cheminement permettant d'arriver à la solution d'un problème décrit par un énoncé

Décrire le cheminement à emprunter pour arriver à votre habitation telle que décrit par votre adresse

# I - Présentation générale

## Le voyageur de commerce

- Un voyageur de commerce désire faire sa tournée, existe-t-il une tournée de moins de 50 km ?



|             |      |
|-------------|------|
| a-b-e-c-d-a | 50km |
|-------------|------|

# I - Présentation générale

---

## •Qu'est ce que l'algorithmique ?

### - Définition

Exprime les instructions résolvant un problème donné **indépendamment des particularités de tel ou tel langage**

→ Apprendre l'algorithmique, c'est apprendre à manier la structure logique d'un programme informatique quel que soit le langage de programmation

### - Intérêt

→ Découpage d'un problème complexe en tâches élémentaires

→ Présentation d'une approche d'un problème compréhensible par des personnes potentiellement non informaticiennes

→ Permet de réaliser un programme juste mais ayant de plus les qualités (lisibilité, portabilité, réutilisabilité, maintenance) et les performances (temps d'exécution, espace mémoire nécessaire) les meilleures possibles

# I - Présentation générale

---

## •Caractéristiques d'un algorithme

### - de haut niveau ("portable")

- doit pouvoir être traduit dans n'importe quel langage de programmation
- ne doit pas faire appel à des notions techniques relatives à un langage particulier ou bien à un SE donné

### - précis

- aucun élément de l'algorithme ne doit porter à confusion

### - concis

- un algorithme ne doit pas dépasser une page sinon il faut décomposer le problème en plusieurs sous-problèmes

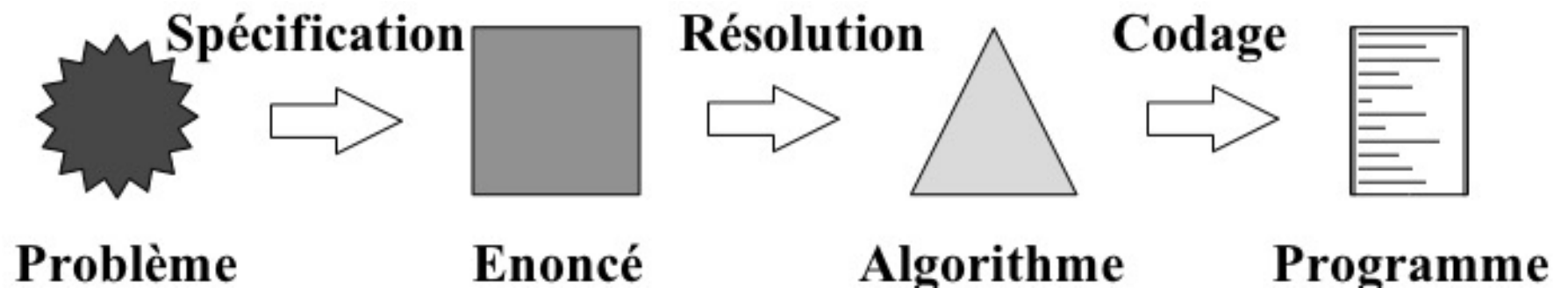
### - structuré

- un algorithme doit être composé de différentes parties facilement identifiables

# I - Présentation générale

## Notions de programme

Spécification d'un schéma de calcul sous forme d'une suite finie d'opérations élémentaires obéissant à un enchaînement déterminé



**Ne pas se laisser aveugler par l'objectif final : le codage !**

# I - Présentation générale

---

## Programme vs algorithme

Un **programme** est donc la description d'un *algorithme* dans un langage accepté par la machine

Un **algorithme**, à l'inverse d'un *programme*, est indépendant du langage de programmation (et donc de ce machine)

# Algorithmique et Structure de données

## *Partie II* *Opérations de base*

## II – Opérations de base

---

### • Représentation algorithmique

#### - Structure générale

→ Nom du programme

Programme <nom\_prog>

→ Déclaration des variables et constantes

Var <liste var> : <type>

Const <liste const> : <type> = <valeur>  
etc

ENTIER, REEL,  
CARACTERE,

→ Début et fin de  
programme

Début

<instructions>

Fin

## II – Opérations de base

---

### •Représentation algorithmique

#### - Déclaration des variables et des constantes

- Toutes les données utilisées dans l'algorithme doivent être déclarées
- Le type d'une donnée spécifie la taille qu'elle occupera en mémoire, l'intervalle des valeurs autorisées et les opérations qui lui seront applicables

Elles peuvent être déclarées :

- comme des variables pour les données dont la valeur sera lue ou modifiée dans le programme (données en E, S ou données intermédiaires)
- comme des constantes pour les données dont la valeur est fixée une fois pour toutes dans le programme (hypothèses)

## II – Opérations de base

---

### **Variables**

**Un algorithme manipule des données stockées dans des variables**

**Chaque variable possède:**

Un nom

Emplacement mémoire

Une adresse

Un type

## II – Opérations de base

---

### Variables

#### Types de données de base

Entiers (12, 789)

Caractères ('a', 'g', '3')

Réels (-7.9; 0,5; 3,14; 1/3)

Chaînes de caractères ("je mange du ...")

Booléens (vrai, faux)

## II – Opérations de base

---

### Déclaration de variables

**Déclaration** d'une variable = réservation d'un emplacement mémoire, auquel on donne un nom unique appelé *identificateur* et par lequel on peut accéder à sa valeur.

### **Syntaxe :**

identificateur : type

### **Exemple**

a : entier

x, y, z : entier

## II – Opérations de base

---

### Constantes

**Une constante est, comme une variable, un emplacement de la mémoire mais sa valeur ne peut pas changer au cours de l'exécution du programme**

**Déclaration d'une constante est toujours associée à son initialisation (première valeur).**

**Syntaxe :**

identificateur : type = valeur

**Exemple**

max : entier = 32767

## II – Opérations de base

### Opérations de l'algorithmique

| Type      | Exemples de valeurs                         | opérations possibles                                                  | opérateur ou mot clé correspondant       |
|-----------|---------------------------------------------|-----------------------------------------------------------------------|------------------------------------------|
| réel      | -15.69 , 0.36                               | addition, soustraction, multiplication, division, comparaison         | +, -, *, /,<br><, ≤, >, ≥, =, ≠          |
| entier    | -10 , 0 , 3 , 689                           | addition, soustraction, multiplication, division, modulo, comparaison | +, -, *, div, mod,<br><, ≤, >, ≥, =, ≠   |
| caractère | 'B' , 'h' , '£' , '?'                       | successeur, prédécesseur, ordre, caractère, comparaison               | suc, pred, ord, car,<br><, ≤, >, ≥, =, ≠ |
| chaîne    | "Bonjour" , "93000",<br>"toto@caramail.com" | longueur, concaténation, comparaison                                  | longueur , + ,<br><, ≤, >, ≥, =, ≠       |
| booléen   | VRAI , FAUX                                 | négation, conjonction , disjonction                                   | NON , ET , OU                            |

## II – Opérations de base

### Comparaison de caractères (1)

Les caractères sont comparés selon l'ordre du code ASCII. Par exemple le caractère '**Z**' (majuscule), de code ASCII **90** est inférieure au caractère '**a**' (minuscule) de code ASCII **97**

Soit la variable **c** de type caractère:

**succ(c)** → caractère suivant **c** selon le code ASCII

**pred(c)** → caractère précédant **c** selon le code ASCII

**ord(c)** → code ASCII du caractère **c**.

**car(n)** → caractère correspondant au code ASCII **n**.

## II – Opérations de base

### Le type booléen

**2 valeurs possibles: VRAI et FAUX**

**Opérateurs de base : ET, OU, NON, OU EX**

**Table de vérité**

| A    | B    | A ET B | A OU B | NON A |
|------|------|--------|--------|-------|
| VRAI | VRAI | VRAI   | VRAI   | FAUX  |
| VRAI | FAUX | FAUX   | VRAI   | FAUX  |
| FAUX | VRAI | FAUX   | VRAI   | VRAI  |
| FAUX | FAUX | FAUX   | FAUX   | VRAI  |

## II – Opérations de base

---

### • Représentation algorithmique

#### - Instructions

→ Lecture et écriture de données

*Lire* *<donnée>, <donnée>*

*Ecrire* *<donnée>, <donnée>*

→ Affectation d'une valeur à une variable

*<variable> ← <valeur>*

*<variable> ← <expression>*

Une expression peut être formée à l'aide de divers opérateurs

→ Opérateurs arithmétiques

changement de signe -

addition +      soustraction -      multiplication \*

division /      division entière : *div*      reste division entière : *mod*

## II – Opérations de base

---

- Représentation algorithmique

- AFFECTATION

**Mettre une valeur dans une variable**

**Représentée par ←**

Variable ← expression

$X \leftarrow 12$

$X \leftarrow 5 + Y$



## II – Opérations de base

---

### Lecture

**permet au programme de lire des données entrées au clavier  
affecte les valeurs entrées au clavier à des variables**

**Syntaxe**

**Lire (*variable<sub>1</sub>, variable<sub>2</sub>, ..., variable<sub>n</sub>*)**

**Lire (x)**

**Lire (x, y)**



## II – Opérations de base

---

### Affichage

#### Afficher des expressions à l'écran

##### Syntaxe

**Ecrire**(expression<sub>1</sub>, expression<sub>2</sub>, ..., expression<sub>n</sub>)

##### Exemples

Écrire ("Bonjour!") → **Bonjour!**

Ecrire (x) → **Daba** si **x="Daba"**

Ecrire (x , "est agée de 19 " , y)

A L'Ecran → **Daba est agée de 19 ans** si **y= "ans"**;

## II – Opérations de base

---

### • Représentation algorithmique

Exemples :

Calcul de TVA

*Programme TVA*

*Var*

*HT, MTVA, TTC : REEL*

*Const*

*TVA : REEL = 19.6*

*Début*

*Lire HT*

*MTVA  $\leftarrow$  HT \* (TVA/100)*

*TTC  $\leftarrow$  HT + MTVA*

*Ecrire TTC, MTVA*

*Fin*

Calcul en binaire

*Programme Binaire*

*Var*

*val1, val2, res : ENTIER*

*Début*

*Lire val1, val2*

*res  $\leftarrow$  val1 OR val2*

*Ecrire res*

*res  $\leftarrow$  val1 AND val2*

*Ecrire res*

*Fin*

## II – Opérations de base

---

### **Algorithme** Echange

#### **variable**

x , y : **entier**

tmp : **entier**

#### **Début**

**Ecrire**("Donner la valeur de l'entier x:")

**Lire**(x)

**Ecrire**("Donner la valeur de l'entier y:")

**Lire**(y)

**Ecrire**("Avant échange: x vaut ",x, "et y vaut ",y)

tmp  $\leftarrow$  x

x  $\leftarrow$  y

y  $\leftarrow$  tmp

**Ecrire**("Après échange: x vaut ",x, "et y vaut ",y)

#### **Fin**