

# Algorithmique de base

Module : Fondement de la programmation

Douglas Teodoro



2019-2020

# SOMMAIRE

## Présentation du cours

Introduction à l'algorithmique

Les fondements de l'informatique

L'algorithmique

Les langages d'implémentation

# PRÉSENTATION DU COURS

# ORGANISATION DU MODULE 631-1

## 2 parties

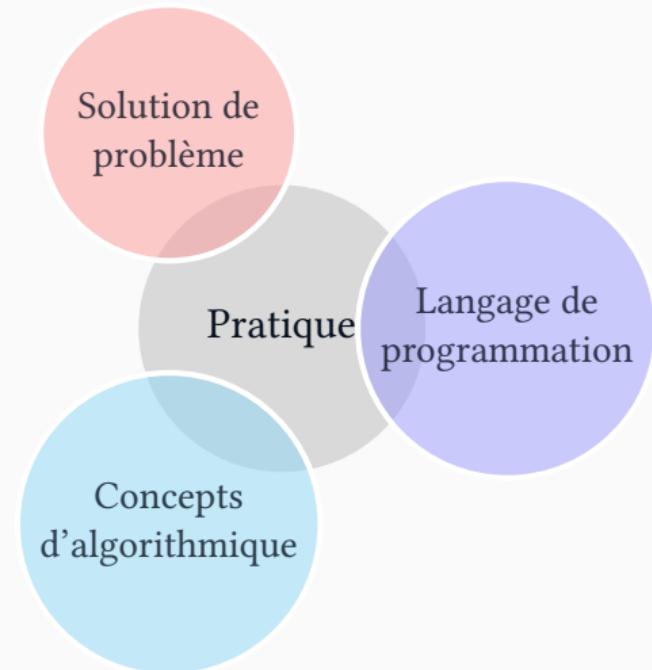
- ▶ Algorithmique de base
- ▶ Introduction au langage Python

## 3 intervenants

- ▶ Jérôme Humbert ([jerome.humbert@hesge.ch](mailto:jerome.humbert@hesge.ch))
- ▶ Sonia Perrote ([sonia.perrote@hesge.ch](mailto:sonia.perrote@hesge.ch))
- ▶ Douglas Teodoro ([douglas.teodoro@hesge.ch](mailto:douglas.teodoro@hesge.ch))

# OBJECTIF DU COURS

- ▶ Maîtriser les **notions fondamentales** de l'algorithmique
- ▶ Exprimer **la solution d'un problème** au moyen de schémas algorithmiques élémentaires
- ▶ Mettre en œuvre les **formants algorithmiques** pour élaborer la solution d'un problème



# PROGRAMME DU COURS

- ▶ **Introduction aux algorithmes**
  - ▶ Généralités et notions de base
  - ▶ Variables, type et codage des données
  - ▶ Affectation et opérateurs
- ▶ **Les structures de contrôle**
  - ▶ La séquence
  - ▶ Les tests et la logique booléenne
  - ▶ Les boucles et leur terminaison
- ▶ **Les sous-programmes**
  - ▶ Procédures et fonctions
  - ▶ Passage de paramètres
  - ▶ La fonction «main»
- ▶ **Les structures de données et fichiers**
  - ▶ Tableaux uni- et multidimensionnels
  - ▶ Tableaux de chaînes de caractères
  - ▶ Traitement de fichiers

# EXPÉRIENCE DES ÉTUDIANT-E-S

Niveau de difficulté rencontré pour : Développer un algorithme pour résoudre un problème (sur papier ou conceptuellement)

RÉPONSE	MOYENNE	TOTAL
1 (Très difficile)	■ 10%	3
2	■ 39%	12
3	■ 32%	10
4	■ 16%	5
5 (Très facile)	■ 3%	1
Total des réponses à la question	■ 100%	31/31

Niveau de difficulté rencontré pour : Comprendre la définition d'un problème

RÉPONSE	MOYENNE	TOTAL
2	■ 35%	11
3	■ 29%	9
4	■ 26%	8
5 (Très facile)	■ 10%	3
Total des réponses à la question	■ 100%	31/31

# MÉTHODES PÉDAGOGIQUES

## Organisation

- ▶ Durée du module : 15 semaines
- ▶ Deux heures de cours théoriques (50%) / pratiques (50%)
- ▶ Deux heures de laboratoire en conjonction avec l'unité « Introduction au langage Python »
- ▶ Un assistant est à disposition dehors de ces séances sur rendez-vous

# MÉTHODES PÉDAGOGIQUES

## Forme

- ▶ **Partie théorique** : n'interdit nullement les questions et la participation des étudiant-e-s
- ▶ **Partie pratique** : repose essentiellement sur la **participation active** des étudiant-e-s
- ▶ **Consacre un temps** à l'étude de ses notes et à la résolution des problèmes (**coder!**)

# MÉTHODES PÉDAGOGIQUES

## Forme

- ▶ **Partie théorique** : n'interdit nullement les questions et la participation des étudiant-e-s
- ▶ **Partie pratique** : repose essentiellement sur la **participation active** des étudiant-e-s
- ▶ **Consacre un temps** à l'étude de ses notes et à la résolution des problèmes (**coder!**)

## Philosophie

L'étudiant-e est encouragé-e à **prendre en charge son propre processus d'apprentissage**

# MODE D'ÉVALUATION

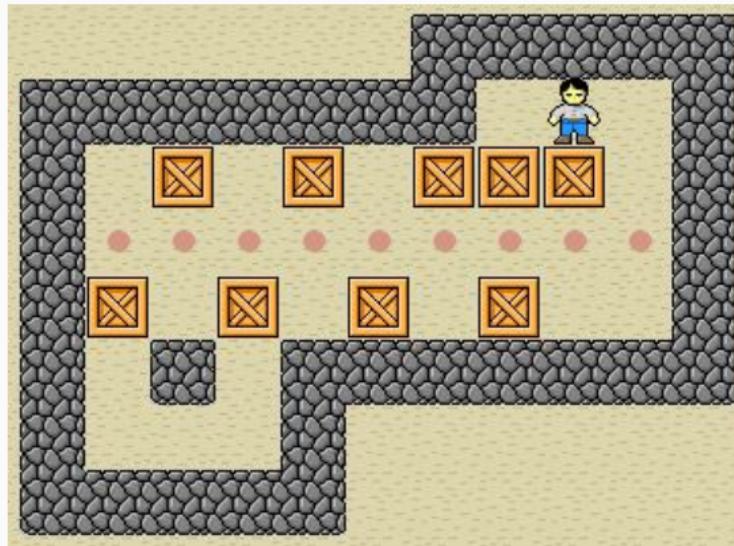
## Contrôle continu - 50%

- ▶ Des travaux pratiques individuels (avec doc) (**15%**)
  - ▶ Rendu : semaines 4, 6, 8, 10, 13
- ▶ Un projet pratique en binôme (**15%**)
  - ▶ Rendu : semaines 14
- ▶ Un contrôle continu pratique individuel (sans doc) d'une durée approx. de 90 minutes (**20%**)
  - ▶ Mercredi 04 décembre 2019 à 17h15 (semaine 11)

## Examen - 50%

- ▶ Un examen écrit et pratique interdisciplinaire
  - ▶ L'examen aura lieu lors de la semaine du 20 janvier 2020 (semaine 16)
  - ▶ Rendu de 75% des travaux pratiques est exigé pour se présenter à l'examen

## PROJET - PYTHON



**Objectif :** développer des algorithmes pour les fonctionnalités du jeu

# MODE D'ÉVALUATION

## Note des unités

- ▶ Moyenne pondérée des notes des contrôles continus
- ▶ note = 20% CC +15% TPs +15% PROJ

## Formation de la note du module

- ▶ Note des contrôles continus : 50%
- ▶ Note de l'examen : 50%

# RÉFÉRENCES

Ebel et Rohaut, *Algorithmique - Techniques fondamentales de programmation* (ENI, 2018)

Bibliothèque numérique ENI, <https://aai-logon.hes-so.ch/eni>

Cormen, *Algorithmes Notions de base* (Dunod, 2013)

ScholarVox, <http://hesge.scholarvox.com>

Sedgewick, Wayne et Dondero et Rohaut, *Introduction to Programming in Python* (Pearson Education, 2015)

Cormen, Leiserson, Rivest et Stein, *Introduction à l'algorithmique* (Dunod, 2010)

Sedgewick et Wayne, *Algorithms* (Pearson Education, 2011)

Deitel et Deitel, *JAVA How to program* (Prentice Hall, 2012)

# CYBERLEAN

- ▶ **Cours :** 19\_HES-SO-GE\_631-1 FONDEMENT DE LA PROGRAMMATION
- ▶ **Mot de passe :** welcome

# SOMMAIRE

Présentation du cours

Introduction à l'algorithmique

Les fondements de l'informatique

L'algorithmique

Les langages d'implémentation

# INTRODUCTION À L'ALGORITHMIQUE

# INFORMATIQUE

## Définition

science du traitement automatique de l'information

## Origine

- ▶ informatik : **automatische Informationsverarbeitung**
- ▶ informatique : **information automatique**

K. Steinbuch, 1957

P. Dreyfus, 1962

# MATÉRIEL ET LOGICIEL

matériel (ordinateur) ensemble de circuits électroniques permettant de manipuler des informations



logiciel (programme) ensemble de instructions remplissant une fonction déterminée, permettant l'accomplissement d'une tâche donnée

```
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
```

```
constructor (nom, prenom, mail) {
    Object.assign(this, {nom, prenom, mail});
    //this.nom = nom;
    //this.mail = mail;
    this.inscription();
}

inscription () {
    console.log(this.nom + " est bien inscrit aux inscriptions");
}

/**
 * Retourne le nom
 * @return {string} - nom
 */
getNom () {
    return (this.constructor.nom);
}

/**
 * Retourne le nom
 * @return {string} - nom
 */
*/
```

# ARCHITECTURE DE VON NEUMANN

## Projet ENIAC

ENIAC, Electronic Numerical Integrator and Computer, 1945

- ▶ lignes essentielles pour construire une machine électronique
- ▶ appliqués jusqu'à nos jours



John Von Neumann,  
mathématicien et  
physicien hongrois

## ARCHITECTURE DE VON NEUMANN



**Instruction per sec**

► ENIAC  
→ 5 milles ( $10^3$ )

► i7  
→

## ARCHITECTURE DE VON NEUMANN



**Instruction per sec**

- ▶ **ENIAC**  
→ 5 milles ( $10^3$ )
- ▶ **i7**  
→ 161 milliards ( $10^9$ )

## LE PROCESSEUR I7 *contre* LA LUMIÈRE

### Question

Qui prendra le plus de temps ?

1. la lumière pour parcourir 100 mètres dans le vacuum  $3 \times 10^8$  m/s
2. le processeur i7 pour exécuter 100 000 instructions  $3 \times 10^5$  MIPS

## LE PROCESSEUR I7 *contre* LA LUMIÈRE

### Question

Qui prendra le plus de temps ?

1. la lumière pour parcourir 100 mètres dans le vacuum

$3 \times 10^8$  m/s

2. le processeur i7 pour exécuter 100 000 instructions

$3 \times 10^5$  MIPS

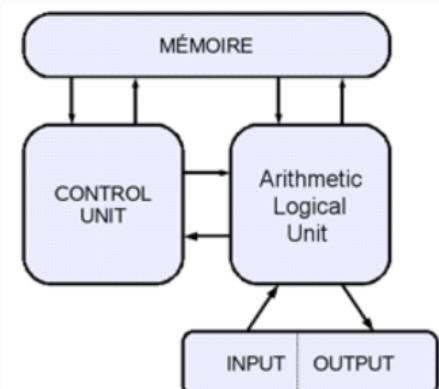
- ▶ lumière :  $3\mu s$
- ▶ processeur i7 :  $3\mu s$



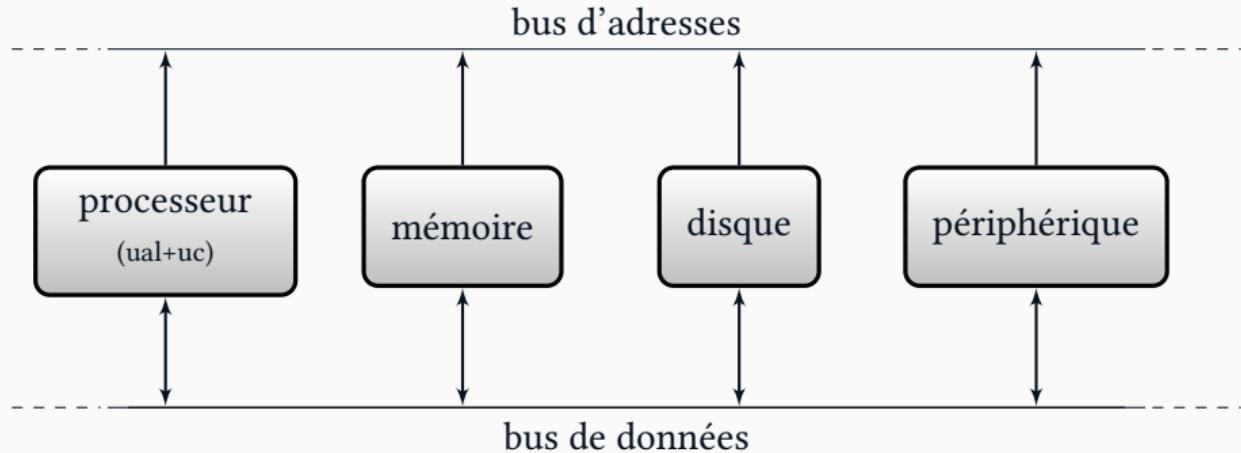
# COMPOSANT DE L'ORDINATEUR

## Architecture de Von Neumann

- ▶ **unité arithmétique et logique (UAL)** : exécute les calculs
- ▶ **unité de contrôle (UC)** : contrôle le séquençage des opérations
- ▶ **mémoire** : stocke des informations (adresse, taille)
- ▶ **entrées/sorties (E/S)** : permettent de communiquer avec le monde extérieur (clavier, écran, etc.)



# ARCHITECTURE DE VON NEUMANN



# L'UNITÉ ARITHMÉTIQUE ET LOGIQUE - UAL (ALU)

Responsable pour les **calculs**

- ▶ unité de traitement arithmétique
- ▶ unité de traitement logique
- ▶ registres

Exemple

- ▶ arithmétique :  $1 + 1 = 2$ ;  $3 \times 2 = 6$ ;  $4.5 / 2.0 = 2.25$ ;
- ▶ logique :  $0 \text{ AND } 1 = 0$ ;  $0 \text{ OU } 1 = 0$
- ▶ registre :  $r1 \leftarrow 5$ ;  $r2 \leftarrow 2.5 + 1.5$

## L'UNITÉ DE CONTRÔLE - UC

**Commande et contrôle** le fonctionnement de l'ordinateur

- ▶ coordonne l'ensemble des tâches
- ▶ est en relation avec la **mémoire** principale
- ▶ est associée au registre à **instruction**

# LA MÉMOIRE

## Stockage d'information

- ▶ lieu de **stockage** permanent ou non de l'information
- ▶ directement reliée à l'**unité centrale**
- ▶ contient le ou les **programmes** à exécuter
- ▶ contient les **données**

1010 1010	adr <sub>1</sub>
0001 1000	adr <sub>2</sub>
1100 1111	adr <sub>3</sub>
1111 0010	adr <sub>4</sub>
0010 0111	adr <sub>5</sub>
0000 0001	adr <sub>6</sub>
0111 1111	adr <sub>7</sub>
1000 1110	adr <sub>8</sub>
...	



# ENTRÉES/SORTIES - OU PÉRIPHÉRIQUES - E/S (I/O)

**Communication** avec le monde extérieur (clavier, écran, etc.)

## Exemple

- ▶ les unités de **visualisation** (écran, moniteur, ...)
- ▶ **disques durs**, supports de mémoire amovibles (stick USB), lecteurs de disquettes (?), ...
- ▶ le **clavier**, la **souris**, le crayon optique
- ▶ les **imprimantes**, les modems (modulateur-démodulateur)



# ARCHITECTURE DE VON NEUMANN

## Déroulement d'un programme

1. l'unité de contrôle (UC) **extrait** une instruction de la mémoire
2. **analyse** l'instruction
3. **recherche** en mémoire les **données** concernées par l'instruction
4. **déclenche** l'opération adéquate sur l'UAL ou l'E/S
5. **range** le résultat dans la mémoire

DEMO - von Neumann

# ARCHITECTURE DE VON NEUMANN

## Déroulement d'un programme

1. l'unité de contrôle (UC) **extrait** une instruction de la mémoire
2. **analyse** l'instruction
3. **recherche** en mémoire les **données** concernées par l'instruction
4. **déclenche** l'opération adéquate sur l'UAL ou l'E/S
5. **range** le résultat dans la mémoire

DEMO - von Neumann

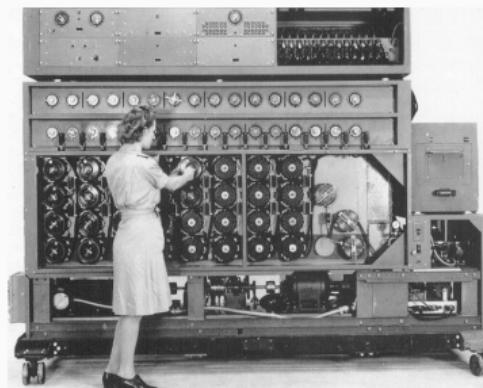
$x = 5$

$y = 3$

$z \leftarrow 13 = 5 + 3 + 5$

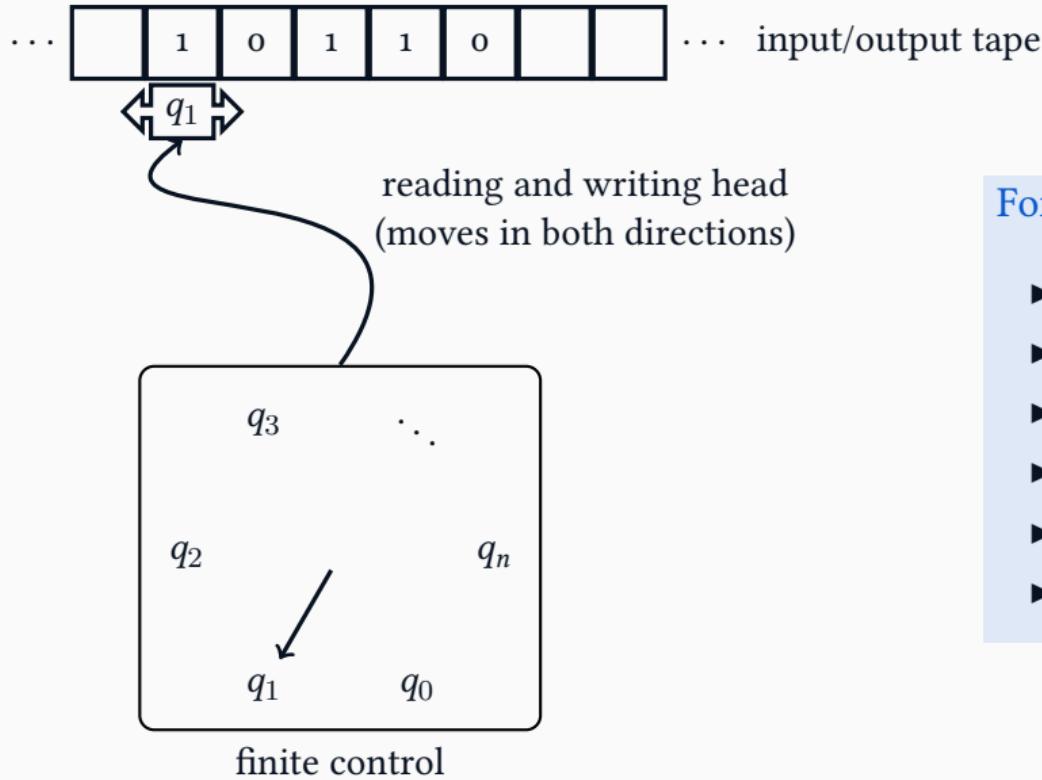
## LA MACHINE DE TURING

- ▶ machine abstraite pour la modélisation du fonctionnement d'un ordinateur
- ▶ inventée pour expliquer la notion de "procédure mécanique"
- ▶ composants : tête de lecture/écriture, ruban infini, état  $q_i$



A. Turing,  
mathématicien et  
cryptologue britannique  
du 20<sup>eme</sup> siècle

# LA MACHINE DE TURING



## Fonctions primitives

- ▶ Lire
- ▶ Ecrire
- ▶ Effacer
- ▶ Déplacer vers la gauche
- ▶ Déplacer vers la droite
- ▶ Arreter

# LA MACHINE DE TURING

DEMO - Machine de Turing

# LA MACHINE DE TURING

**DEMO - Machine de Turing**

$$11_2 = 3_{10} \rightarrow 11_2 + 1_2 = 100_2 = 4_{10}$$

# REPRÉSENTATION DES INSTRUCTIONS ET DES DONNÉES

## L'organisation de la mémoire

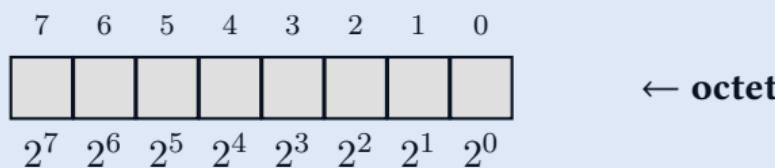
- ▶ **binaire** : représentation de chiffres utilisés par les ordinateurs
- ▶ **base 2** : 0 ou 1
- ▶ plus petite unité d'information : **bit** (contraction de *binary digit*)
- ▶ suivant représenté par groupe de 8 bits : **octet** (*byte*)

## Exemple

Mémoire de  $n$  octets

1	0	1	1	0	0	0	1	adr <sub>1</sub>
0	0	1	1	0	1	0	1	adr <sub>2</sub>
0	0	0	0	1	1	1	1	adr <sub>3</sub>
1	1	1	1	0	0	0	0	adr <sub>4</sub>
...								
1	1	1	1	1	1	1	1	adr <sub><math>n-3</math></sub>
1	0	0	1	1	1	0	1	adr <sub><math>n-2</math></sub>
1	0	0	1	1	1	0	1	adr <sub><math>n-1</math></sub>
1	0	0	1	1	1	0	1	adr <sub><math>n</math></sub>

## SYSTÈME DE NUMÉRATION BINAIRE



→ le bit en position 0 est le **bit de poids faible** et le bit en position 7 est le **bit de poids fort**

## SYSTÈME DE NUMÉRATION BINAIRE

7	6	5	4	3	2	1	0
0	0	0	0	0	1	0	1
$2^7$	$2^6$	$2^5$	$2^4$	$2^3$	$2^2$	$2^1$	$2^0$

← octet

→ le bit en position 0 est le **bit de poids faible** et le bit en position 7 est le **bit de poids fort**

$$\begin{aligned}101_2 &= 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 \\&= 5_{10}\end{aligned}$$



# INFORMATION BINAIRE

## Circuit en série - Logique ET

X	Y	courant	binaire
éteint	éteint	ne passe pas	00
éteint	<b>allumé</b>	ne passe pas	01
<b>allumé</b>	éteint	ne passe pas	10
<b>allumé</b>	<b>allumé</b>	<b>passee</b>	11

## INFORMATION BINAIRE

### Circuit en série - Logique ET

X	Y	courant	binaire
éteint	éteint	ne passe pas	00
éteint	<b>allumé</b>	ne passe pas	01
<b>allumé</b>	éteint	ne passe pas	10
<b>allumé</b>	<b>allumé</b>	<b>passee</b>	11

### Circuit en parallèle - Logique OU

X	Y	courant	binaire
éteint	éteint	ne passe pas	00
éteint	<b>allumé</b>	<b>passee</b>	01
<b>allumé</b>	éteint	<b>passee</b>	10
<b>allumé</b>	<b>allumé</b>	<b>passee</b>	11

# QCM 1

## L'informatique est la science

- A) des dispositifs dont le fonctionnement dépend de la circulation d'électrons
- B) des signaux électriques porteurs d'information ou d'énergie
- C) du traitement automatique de l'information
- D) de la commande des appareils fonctionnant sans intervention humaine

# QCM 1

## L'informatique est la science

- A) des dispositifs dont le fonctionnement dépend de la circulation d'électrons
- B) des signaux électriques porteurs d'information ou d'énergie
- C) du traitement automatique de l'information
- D) de la commande des appareils fonctionnant sans intervention humaine

## QCM 2

**Un bit est**

- A) un chiffre binaire
- B) composé de 8 chiffres binaires
- C) un chiffre héxadécimal
- D) un mot d'un langage informatique

## QCM 2

**Un bit est**

- A) un chiffre binaire
- B) composé de 8 chiffres binaires
- C) un chiffre héxadécimal
- D) un mot d'un langage informatique

## QCM 3

**Sans tenir compte du signe, quelle est la valeur maximale d'un nombre codé en 8 bits (octet)?**

- A) 256
- B) 16
- C) 255
- D) 8

## QCM 3

**Sans tenir compte du signe, quelle est la valeur maximale d'un nombre codé en 8 bits (octet)?**

- A) 256
- B) 16
- C) 255
- D) 8

$$2^8 - 1 = 2^7 + 2^6 + 2^5 + 2^4 + 2^3 + 2^2 + 2^1 + 2^0 = 255_{10}$$

## QCM 3

**Sans tenir compte du signe, quelle est la valeur maximale d'un nombre codé en 8 bits (octet)?**

- A) 256
- B) 16
- C) 255
- D) 8

$$2^8 - 1 = 2^7 + 2^6 + 2^5 + 2^4 + 2^3 + 2^2 + 2^1 + 2^0 = 255_{10}$$

**Et en 64 bits ?**

## QCM 3

**Sans tenir compte du signe, quelle est la valeur maximale d'un nombre codé en 8 bits (octet)?**

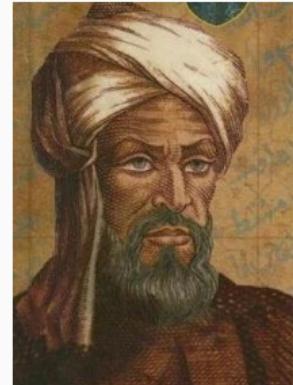
- A) 256
- B) 16
- C) 255
- D) 8

$$2^8 - 1 = 2^7 + 2^6 + 2^5 + 2^4 + 2^3 + 2^2 + 2^1 + 2^0 = 255_{10}$$

**Et en 64 bits ?**

# L'ALGORITHME

**algorithme** méthode de calcul qui indique la **démarche à suivre** pour résoudre une série de problèmes équivalents en appliquant dans un **ordre précis** une **suite finie** de règles



Al-Khwarizmi,  
mathématicien persan  
du 9<sup>ème</sup> siècle

→ un **ensemble d'opérations** de calcul élémentaires qui sont organisées de façon à produire un **ensemble de valeurs en sortie** (output) à partir d'un **ensemble de valeurs fournies en entrée** (input)

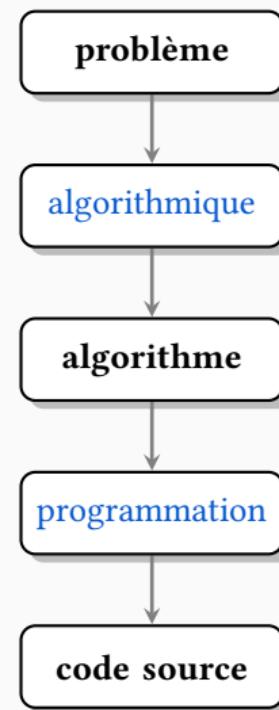
# L'ALGORITHMIQUE

## Definition

algorithmique la science des algorithmes

## Aspects

- ▶ art de design et construction
- ▶ théorie de complexité
- ▶ analyse de validité, robustesse, réutilisabilité



## L'ALGORITHMIQUE

Un **algorithme** exprime la structure logique d'un programme :  
⇒ il est **indépendant** du langage de programmation et de l'ordinateur



## Ada Lovelace,

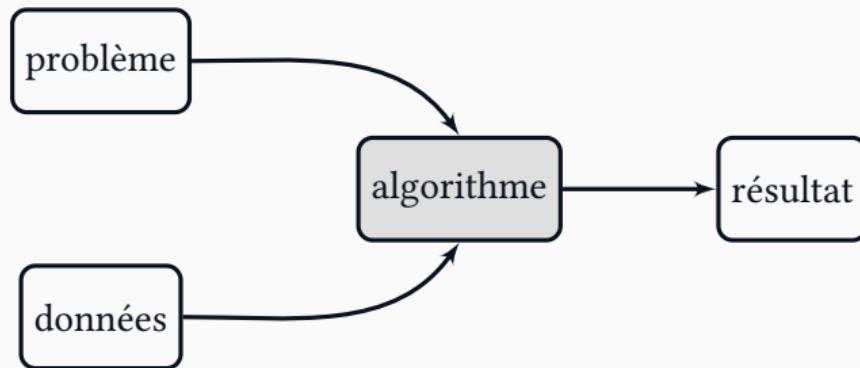
mathématicienne britannique qui a écrit le 1<sup>er</sup> programme d'ordinateur au 19<sup>ème</sup> siècle

## L'ALGORITHMIQUE - ADA LOVELACE 1ER PROGRAMME

**Diagram for the computation by the Engine of the Numbers of Bernoulli.** See Note G. (page 10)

## ALGORITHME ET PROBLÈME

**Algorithm** : prendre en **entrée** la description d'un **problème** et une **instance** de ce problème, c'est-à-dire des **données**, et fournira en **sortie** une solution à l'instance du problème examiné :



**Exemple** : calculer la moyenne (problème) de 3 et 5 (instance) → 4 (résultat)

## PROBLÈMES ET INSTANCES

Un **problème** ne sera véritablement bien spécifié que s'il s'inscrit dans le schéma suivant :

étant donné [**les données**], on demande [**le résultat attendu**]

La **première étape** dans la résolution d'un problème est de **préciser** ce problème à partir d'un énoncé (souvent flou)

# PROBLÈMES ET INSTANCES

## Exemple : un système de navigation GPS

problème typique étant donné

[deux villes et un réseau routier],  
on demande

[le chemin le plus court possible entre  
ces deux villes]



# PROBLÈMES ET INSTANCES

## Exemple : un système de navigation GPS

problème typique étant donné

[deux villes et un réseau routier],  
on demande

[le chemin le plus court possible entre  
ces deux villes]



instance du problème triplé

[deux villes et un réseau routier]  
⇒ valeurs spécifiques pour le problème

# PROBLÈMES ET INSTANCES

## Exemple : un système de navigation GPS

problème typique étant donné

[deux villes et un réseau routier],  
on demande

[le chemin le plus court possible entre  
ces deux villes]

instance du problème triplé

[deux villes et un réseau routier]  
⇒ valeurs spécifiques pour le problème



Instance :

- ▶ départ = Genève
- ▶ arrivée = Bâle
- ▶ réseau = réseau routier suisse

## PROBLÈMES ET INSTANCES

On adoptera souvent le **format** suivant pour **décrire les problèmes** à résoudre :

### Nom du problème

exemple 1 : Recherche d'itinéraire

exemple 1 : Calcul de la moyenne

### Données

: les données sur lesquelles on travaille

⇒ exemple 1 : une ville de départ, une ville d'arrivée, et un réseau routier

⇒ exemple 2 : une liste des valeurs réels

### Résultat

: l'objectif du problème donné

⇒ exemple 1 : le plus court chemin dans le réseau donné entre les villes données

⇒ exemple 2 : la moyenne des valeurs

## QMC 5

### **L'algorithmique est la science**

- A) du traitement automatique de l'information
- B) des algorithmes
- C) des langages de programmation
- D) des instructions

## QMC 5

### L'algorithmique est la science

- A) du traitement automatique de l'information
- B) des algorithmes
- C) des langages de programmation
- D) des instructions

## QMC 6

### Un algorithme est

- A) un ensemble de programmes remplissant une fonction déterminée, permettant l'accomplissement d'une tâche donnée
- B) une suite ordonnée d'instructions qui indique la démarche à suivre pour résoudre une série de problèmes équivalents
- C) le nombre d'instructions élémentaires à exécuter pour réaliser une tâche donnée
- D) un ensemble de dispositifs physiques utilisés pour traiter automatiquement des informations

## QMC 6

### Un algorithme est

- A) un ensemble de programmes remplissant une fonction déterminée, permettant l'accomplissement d'une tâche donnée
- B) une suite ordonnée d'instructions qui indique la démarche à suivre pour résoudre une série de problèmes équivalents
- C) le nombre d'instructions élémentaires à exécuter pour réaliser une tâche donnée
- D) un ensemble de dispositifs physiques utilisés pour traiter automatiquement des informations

## CARACTÉRISTIQUES D'UN ALGORITHME

### Tout algorithme possède

- un **nom** et s'exprime dans un **langage** (français, anglais, dessins, Python, Java, ...)

## CARACTÉRISTIQUES D'UN ALGORITHME

### Tout algorithme possède

- ▶ un **nom** et s'exprime dans un **langage** (français, anglais, dessins, Python, Java, ...)
- ▶ des **données** (ex., les deux nombres à multiplier pour la multiplication)

## CARACTÉRISTIQUES D'UN ALGORITHME

### Tout algorithme possède

- ▶ un **nom** et s'exprime dans un **langage** (français, anglais, dessins, Python, Java, ...)
- ▶ des **données** (ex., les deux nombres à multiplier pour la multiplication)
- ▶ un ou des **résultats** (ex., le résultat de la multiplication)

## CARACTÉRISTIQUES D'UN ALGORITHME

### Tout algorithme possède

- ▶ un **nom** et s'exprime dans un **langage** (français, anglais, dessins, Python, Java, ...)
- ▶ des **données** (ex., les deux nombres à multiplier pour la multiplication)
- ▶ un ou des **résultats** (ex., le résultat de la multiplication)
- ▶ une **série chronologique** d'étapes ou **sous-algorithmes** lesquels agissent sur les données  
(ex., multiplier le nombre du haut par chacun des chiffres du bas, additionner tous les résultats pour la multiplication)

## CARACTÉRISTIQUES D'UN ALGORITHME

### Tout algorithme possède

- ▶ un **nom** et s'exprime dans un **langage** (français, anglais, dessins, Python, Java, ...)
- ▶ des **données** (ex., les deux nombres à multiplier pour la multiplication)
- ▶ un ou des **résultats** (ex., le résultat de la multiplication)
- ▶ une **série chronologique** d'étapes ou **sous-algorithmes** lesquels agissent sur les données  
(ex., multiplier le nombre du haut par chacun des chiffres du bas, additionner tous les résultats pour la multiplication)
- ▶ certaines phrases justifient ou expliquent ce qui se passe : ce sont des **commentaires**

# QUEL LANGAGE ?

Python `print("Bonjour !")`

JAVA `System.out.print("Bonjour !")`

JavaScript `alert("Bonjour !")`

C++ `std::cout << "Bonjour !"`

R `print("Bonjour !")`

Pascal `WriteLn("Bonjour !")`

Perl `print "Bonjour !"`

Bash `echo "Bonjour !"`

...

# QUEL LANGAGE ?

Python `print("Bonjour !")`

JAVA `System.out.print("Bonjour !")`

JavaScript `alert("Bonjour !")`

C++ `std::cout << "Bonjour !"`

R `print("Bonjour !")`

Pascal `WriteLn("Bonjour !")`

Perl `print "Bonjour !"`

Bash `echo "Bonjour !"`

...

# QUEL LANGAGE ?

## Pseudo-code

Python `print("Bonjour !")`

JAVA `System.out.print("Bonjour !")`

JavaScript `alert("Bonjour !")` écrire "Bonjour !"

C++ `std::cout << "Bonjour !"`

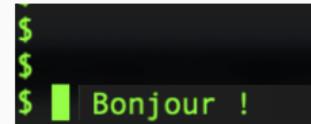
R `print("Bonjour !")`

Pascal `WriteLn("Bonjour !")`

Perl `print "Bonjour !"`

Bash `echo "Bonjour !"`

...



A terminal window with a black background and white text. It shows three dollar signs (\$) on the left, followed by a green vertical bar (representing a cursor), and the text "Bonjour !" in white.

## PSEUDO-CODE OU LANGAGE ALGORITHMIQUE

**pseudo-code** représentation textuelle d'un algorithme  
dit aussi Langage de Description des Algorithmes (LDA)

## PSEUDO-CODE OU LANGAGE ALGORITHMIQUE

pseudo-code représentation textuelle d'un algorithme  
dit aussi Langage de Description des Algorithmes (LDA)

C'est un **langage informel** et **symbolique** qui utilise

- ▶ des **noms symboliques** destinés à représenter les objets sur lesquels s'effectuent des actions
  - ▶ variables
- ▶ des **mots-clés** et des **opérateurs** qui traduisent les opérations exécutables par un exécutant donné
  - ▶ mots-clés : si, tant que, pour, etc.
  - ▶ opérateurs : +, -, \*, etc.

# STRUCTURE DU PSEUDO-CODE OU PYTHON

## Pseudo-code

### Algorithme : Nom du algorithm

**Données** : Données d'entrée

**Résultat** : Résultat attendu

// déclaration des variables

```
1  
2 ...  
3 ...  
4  
5 ...  
6 ...  
7 ...  
8  
9 ...  
10 ...  
11 ...  
12
```

// séquence d'opérations

## Python

```
""" nom du algoritm  
données : ...  
résultat : ...  
"""\n\n### déclaration des variables  
...  
...  
  
### initialisation des variables  
...  
...  
  
### séquence d'opérations  
...  
...
```

# MON 1ER ALGORITHME

## Pseudo-code

### Algorithme : Bienvenue

**Données :** le semestre d'entrée

**Résultat :** affiche un message de bienvenue aux étudiant-e-s

```
// déclaration des variables
```

```
// initialisation des variables
```

```
// séquence d'opérations
```

1

2 afficher "Bonjour !"

3 afficher "Bienvenue aux étudiant-e-s  
2019-2020 !"

4

## Python

```
""" bienvenue
```

```
données : le semestre d'entrée
```

```
résultat : affiche un message de bienvenue  
aux étudiant-e-s
```

```
"""
```

```
### déclaration des variables
```

```
### initialisation des variables
```

```
### séquence d'opérations
```

```
print("Bonjour !")
```

```
print("Bienvenue aux étudiant-e-s 2019-  
2020 !")
```

## EXERCICE

1. En utilisant un [éditeur de texte](#) :
  - 1.1 créez un nouveau fichier
  - 1.2 écrivez l'algorithme « bienvenue » comme dans l'exemple
  - 1.3 sauvegardez avec le nom bienvenue.py
2. En utilisant la [ligne de commande](#) (Initialiser → cmd) :
  - 2.1 allez dans le répertoire où vous avez sauvégarde votre algorithme :  
`cd <chemin\du\repertoire>`
  - 2.2 exécutez le programme/algorithme avec la commande :  
`python bienvenue.py`
3. Modifiez votre algorithme pour qu'il donne une message de bienvenue aux étudiant-e-s de la [filière Informatique de gestion](#)

## ELEMENTS D'UN ALGORITHME

**Algorithme** : résout toujours un problème en utilisant des opérations de calcul élémentaire et des structures de contrôle

## ELEMENTS D'UN ALGORITHME

**Algorithme** : résout toujours un problème en utilisant des opérations de calcul élémentaire et des structures de contrôle

## Opérations de calcul élémentaires

- ▶ opérations arithmétiques + - \* / %
  - ▶ lecture/affectation de variables ←→
  - ▶ comparaisons == != < > ≤≥

## ELEMENTS D'UN ALGORITHME

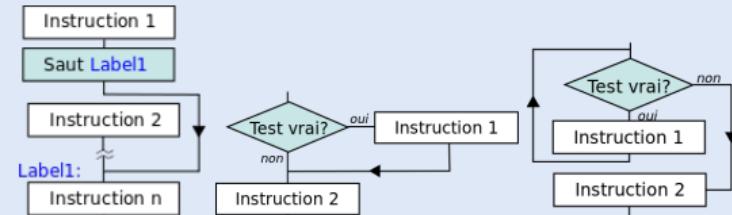
**Algorithme** : résout toujours un problème en utilisant des opérations de calcul élémentaire et des structures de contrôle

## Opérations de calcul élémentaires

- opérations arithmétiques + - \* / %
  - lecture/affectation de variables ←→
  - comparaisons == != < > ≤ ≥

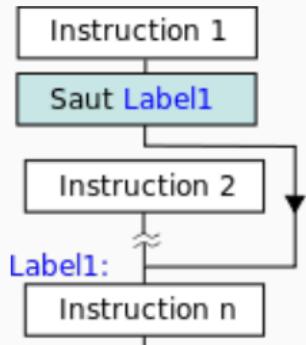
## Structures de contrôle

- ▶ sequence
  - ▶ conditions
  - ▶ boucles

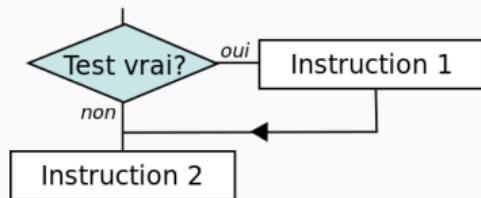


# ELEMENTS D'UN ALGORITHME... COMING

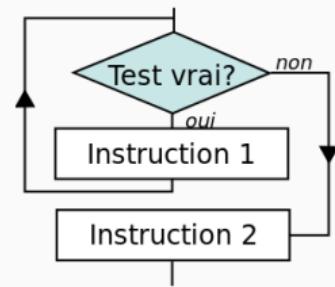
## Séquence



## Condition



## Boucles



# APPLICATIONS PRATIQUES DE L'ALGORITHMIQUE

## Exemple

- ▶ Séquençage du génome humain
- ▶ Internet
  - ▶ Chemin le plus court
  - ▶ Table de Hachage
  - ▶ Filtrage de chaîne de caractère
- ▶ Commerce électronique
  - ▶ Cryptographie
- ▶ Télécommunications
  - ▶ Routage
- ▶ Domotique
  - ▶ Assistance aux personnes âgées ou dépendantes
  - ▶ Minimisation des coûts



## OBJECTIFS DU COURS D'ALGORITHMIQUE DE BASE

- ▶ Acquérir les **notions fondamentales** et travailler la **logique algorithmique**
- ▶ **Apprentissage** d'un langage algorithmique appliqué à la programmation
- ▶ Maîtriser les **concepts de base algorithmique**
  - ▶ Instructions de contrôle
  - ▶ Procédures et fonctions
  - ▶ Structures de données élémentaires
- ▶ **Concevoir** des algorithmes simples

## RÉFÉRENCE

**Algorithmique - Techniques fondamentales de programmation**

Chapitre : Introduction à l'algorithmique

Ebel et Rohaut, <https://aai-logon.hes-so.ch/eni>

**Cyberlearn** : HES-SO-GE\_631-1 Fondements de la programmation

(*welcome*)

<http://cyberlearn.hes-so.ch>