

## **Information, Calcul et Communication**

### **CS-119(k) ICC – Théorie Semaine 1**

Rafael Pires  
[rafael.pires@epfl.ch](mailto:rafael.pires@epfl.ch)

# Pourquoi un cours d'introduction à l'informatique pour SIE et CGC ?

- **4e pilier** de la culture (après la lecture, l'écriture et l'arithmétique)
- Elle constitue désormais une **discipline scientifique à part entière** : la science du traitement automatique de l'information.
- L'informatique a non seulement changé notre société, mais aussi **notre façon de faire de la science**.
- De nos jours, tout·e ingénieur·e qui maîtrise les sciences du numérique a clairement un avantage sur les autres...

# ICC



Information



Calcul



Communication

# ICC

Information



Calcul



Communication



Données

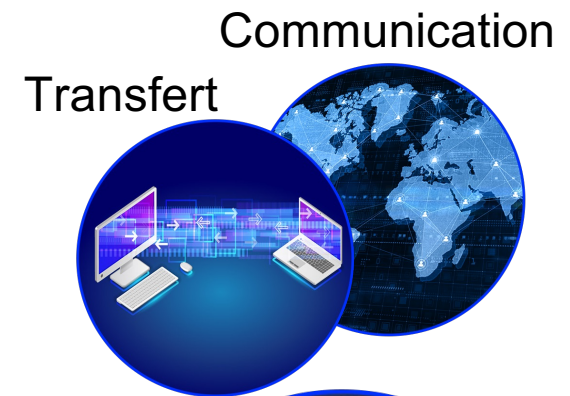
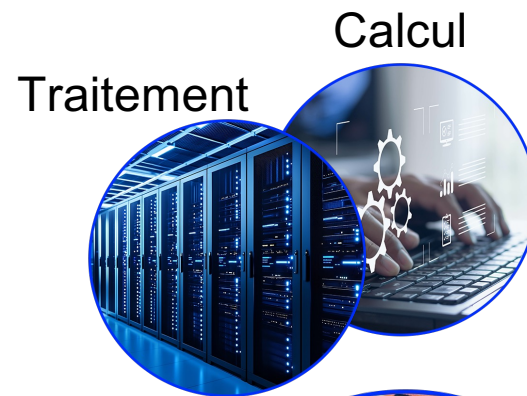


Traitement

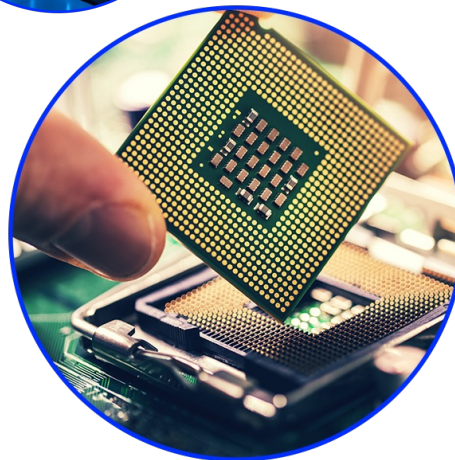


Transfert

# ICC



Stockage



Processeur



Réseau



# ICC

## Information



Données

Stockage

## Calcul



Traitement

Processeur

## Communication



Transfert

Réseau

# Partie théorique (logistique)

## Cours :

- Les vendredis après-midi de 14h15 à 16h, en salle [CE 12](#)
- Pas de diffusion en temps réel, enregistrement disponible au plus tard le lendemain.

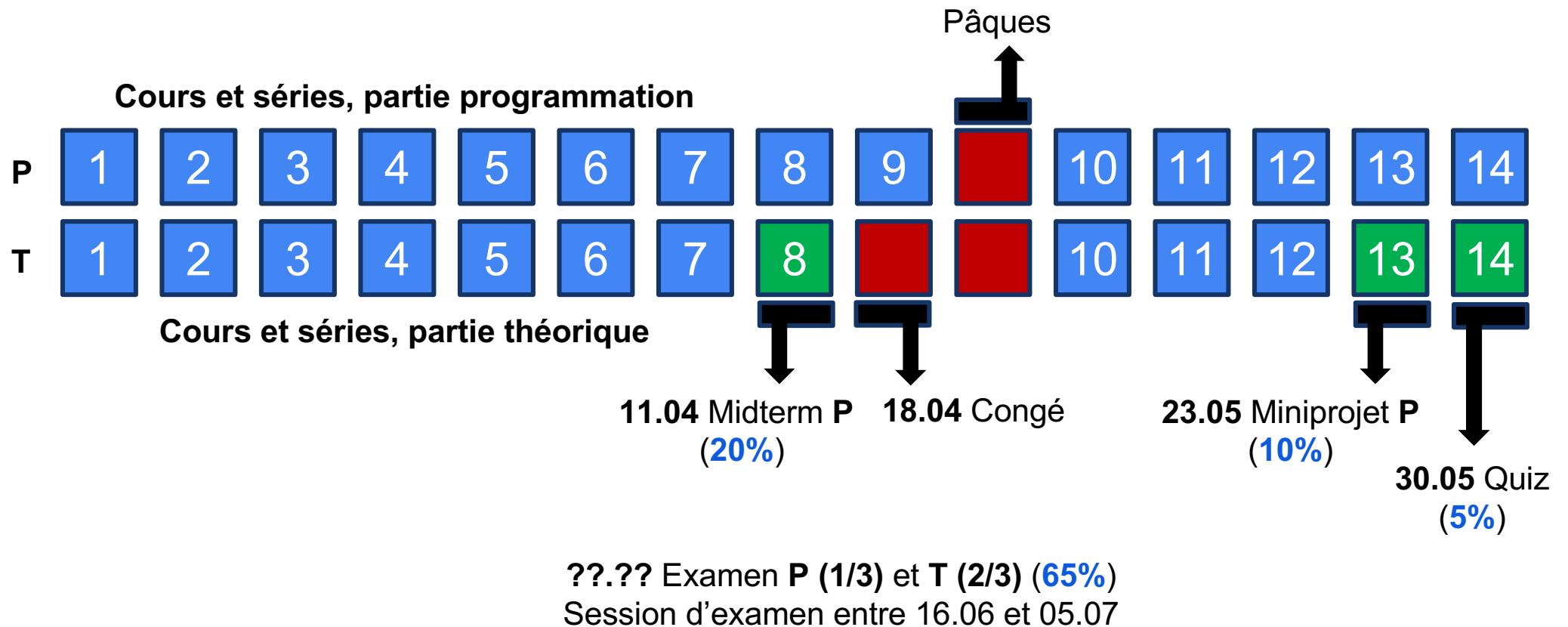
## Exercices :

- Séances d'exercices les vendredis de 16h15 à 17h15 en salles [INF 1](#) et [INF 119](#).

## Références (liens sur Moodle) :

- Questions : Ed discussion
- Livre «Découvrir le numérique», EPFL Press, 2016
- Vidéos sur [mediaspace.epfl.ch](https://mediaspace.epfl.ch) et MOOC sur [courseware.epfl.ch](https://courseware.epfl.ch)

# Programme du cours





# Programme du cours

## Cours et séries, partie programmation

P	1	2	3	4	5	6	7	8	9		10	11	12	13	14
T	1	2	3	4	5	6	7	8			10	11	12	13	14

## Cours et séries, partie théorique



Calcul



Information



Communication<sub>9</sub>

# Programme du cours



## Calcul

- Algorithmes
- Complexité
- Conception d'algorithmes
- Calculabilité
- Circuits



## Information

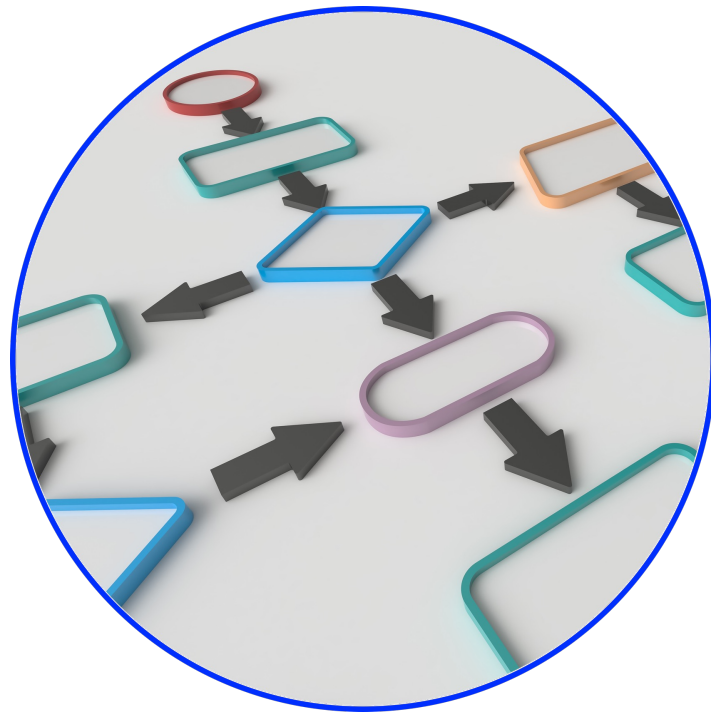
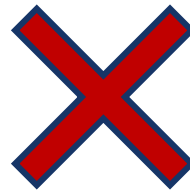
- Représentation de nombres
- Echantillonnage et reconstruction de signaux
- Entropie
- Compression



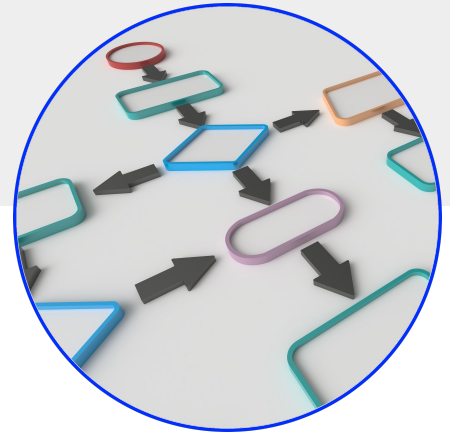
## Communication

- Réseau
- Cryptographie

# Programme vs. algorithmes

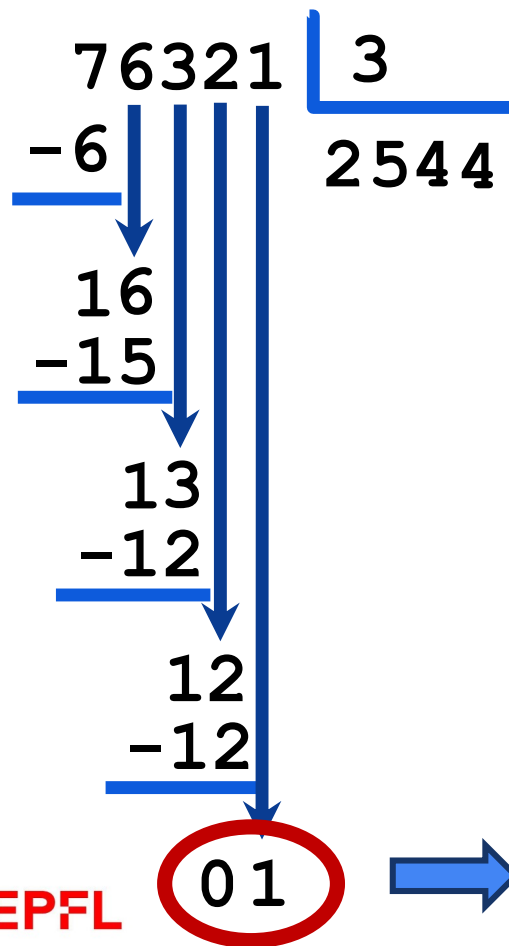


# Qu'est-ce qu'un algorithme ?



- Un algorithme n'est **pas** un programme.
- Un algorithme est la description des étapes **élémentaires** menant à la résolution d'un problème; c'est donc la description conceptuelle d'un programme.
- Un **programme** est l'implémentation d'un algorithme dans un langage donné et dans un système particulier.

# Exemple 1 : calcul du modulo 3 d'un grand nombre



Rappel :

Modulo : le reste  $r$  de la division d'un entier  $a$  par un entier  $b$  non nul.  
 $a \bmod b = r$ , si  $a = q.b + r$  et  $0 \leq r < |b|$

$$7 + 6 + 3 + 2 + 1 = 19$$

$$1 + 9 = 10$$

$$1 + 0 = 1$$

Pourquoi ?

$$47 = 4 \cdot 10 + 7 = 4(1 + \cancel{8}) + 7$$

$$4 + 7 = 11 = 1(1 + \cancel{8}) + 1$$

$$2 \rightarrow 47 \bmod 3 = 2$$

$$\rightarrow 76321 \bmod 3 = 1$$

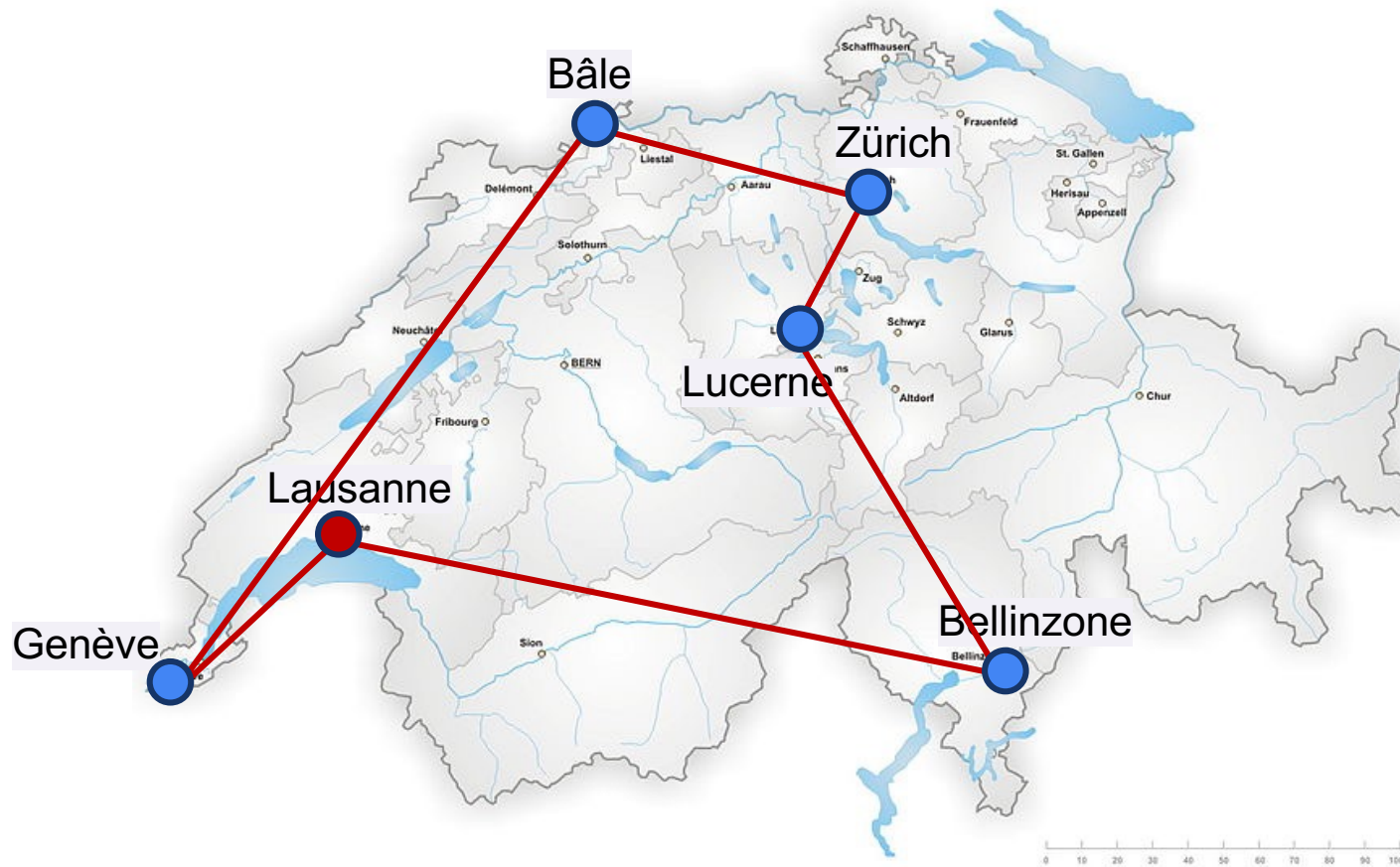
## Exemple 2 : recherche du minimum dans une liste

$$\left\{ \begin{array}{l} L = (13, 47, 18, 15, 11, 19, 46, 18, 15) \\ n = 9 \end{array} \right.$$

1. On considère le premier nombre de la liste le '**minimum**'
2. On le compare au 2ème nombre, le '**suivant**'
3. Si **suivant** < **minimum**, alors **minimum** ← **suivant**
4. Sinon, on continue, c'est à dire, **suivant** ← celui d'après
5. On revient à l'étape 3 jusqu'à la fin de la liste
6. Le résultat est la valeur de '**minimum**'



## Exemple 3 : problème du voyageur de commerce

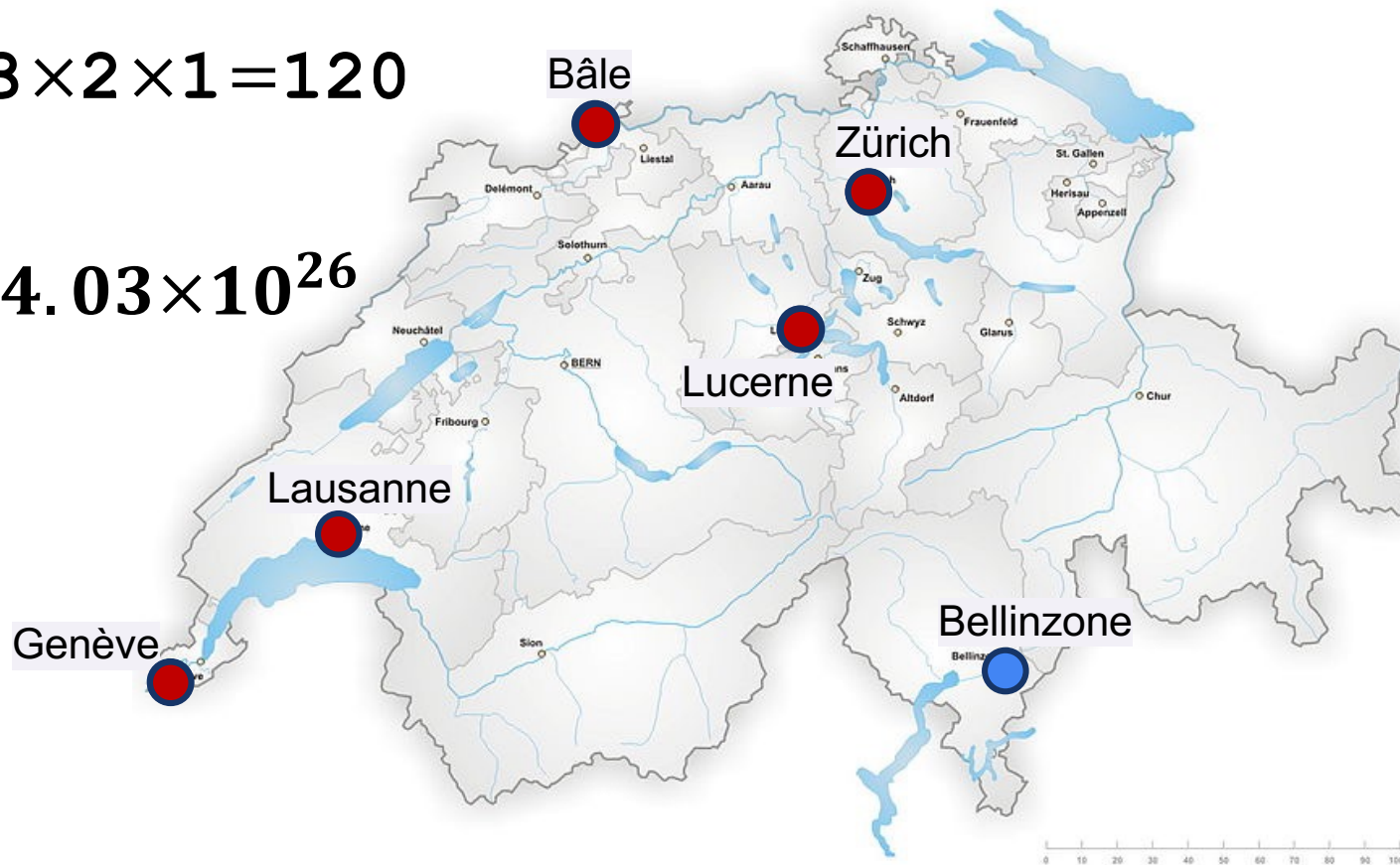


## Exemple 3 : problème du voyageur de commerce

$$5 \times 4 \times 3 \times 2 \times 1 = 120$$

$n!$

$$26! \approx 4.03 \times 10^{26}$$



Factoriale	Résultat
1!	1
2!	2
3!	6
4!	24
5!	120
6!	720
7!	5040
8!	40320
9!	362880
10!	3628800
11!	39916800
12!	479001600
13!	6227020800
14!	87178297200
15!	1307674368000
16!	20922789888000
17!	355687428096000
18!	6402373705728000
19!	121645100408832000
20!	2432902008176640000
21!	51090942171709440000
22!	1124000727777607680000
23!	25852016738884976640000
24!	620448401733239439360000
25!	15511210043330985984000000

# Algorithmes : ingrédients de base

Données



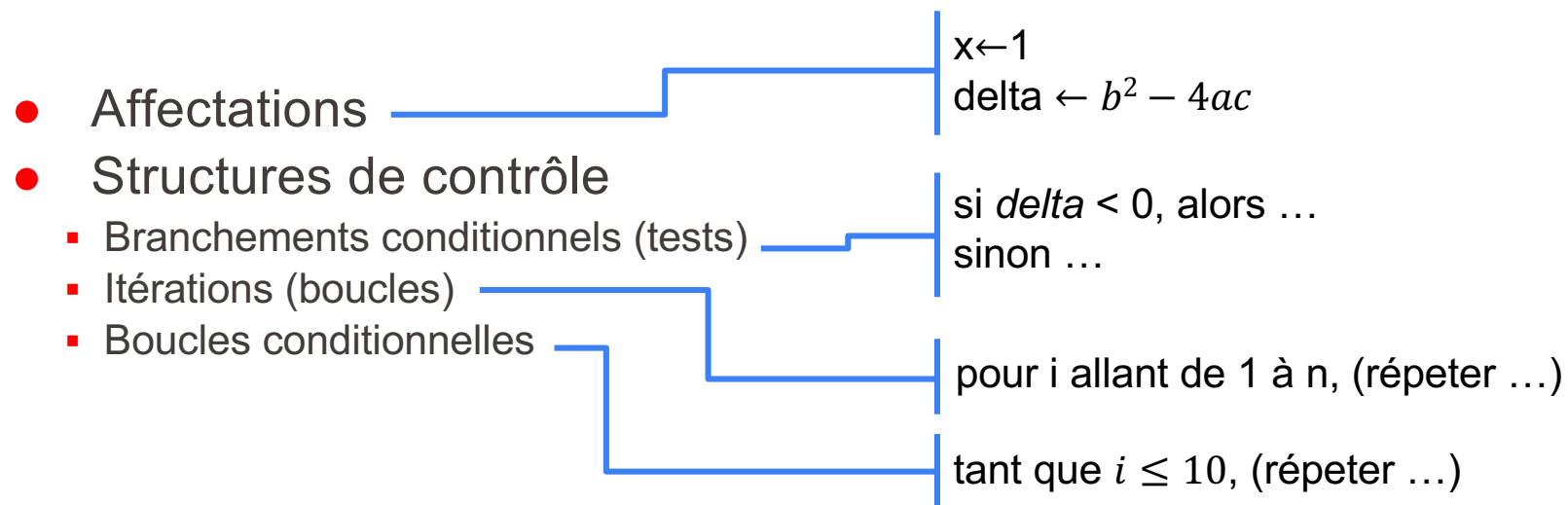
- Entrées
- Sorties
- Variables internes

Traitement



- Affectations
- Structures de contrôle
  - Branchements conditionnels (tests)
  - Itérations (boucles)
  - Boucles conditionnelles

# Algorithmes : instructions



# Ingrédients de base

## Données

- Entrées
- Sorties
- Variables internes

## Instructions

- Affectations
- Structures de contrôle
  - Branchements conditionnels (tests)
  - Itérations (boucles)
  - Boucles conditionnelles

## Entrées

$$\begin{cases} L = (13, 47, 18, 15, 11, 19, 46, 18, 15) \\ n = 9 \end{cases}$$

## Variables internes

Branchement  
conditionnel

Boucle  
conditionnel

1. On considère le premier nombre de la liste le '**minimum**'
2. On le compare au 2ème nombre, le '**suivant**'
3. Si **suivant** < **minimum**, alors **minimum** ← **suivant**
4. Sinon, on continue, c'est à dire, **suivant** ← celui d'après
5. On revient à l'étape 3 jusqu'à la fin de la liste
6. Le résultat est la valeur de '**minimum**'

Affectations

Sortie

# Pseudo-code

$$\begin{cases} L = (13, 47, 18, 15, 11, 19, 46, 18, 15) \\ n = 9 \end{cases}$$

1. On considère le premier nombre de la liste le '**minimum**'
2. On le compare au 2ème nombre, le '**suivant**'
3. Si **suivant** < **minimum**, alors **minimum** ← **suivant**
4. Sinon, on continue, c'est à dire, **suivant** ← celui d'après
5. On revient à l'étape 3 jusqu'à la fin de la liste
6. Le résultat est la valeur de '**minimum**'

## Valeur minimale

**entrée** : liste **L** de nombres entiers, de taille **n**  
**sortie** : la (ou une des) valeur(s) minimale(s) de la liste

```
min ← L(1)
Pour i allant de 2 à n
    Si L(i) < min
        min ← L(i)
Sortir : min
```



# Comparaison tous contre tous

- **Question**

- Est-ce que tous les objets visibles sur cette photo sont différents les uns des autres ?

- **Question réciproque**

- Y a-t-il au moins deux objets identiques sur cette photo ?



# Tous les 3 différents ?

- **Problème à résoudre :**

- Parmi une liste de 3 objets, identifier si ceux-ci sont tous différents les uns des autres.



## Tous les 3 différents

**entrée** : liste **L** de 3 objets  
**sortie** : valeur binaire oui/non

**s** ← **oui**  
Pour **i** allant de 1 à 3 :  
    Pour **k** allant de 1 à 3 :  
        Si **L(i) = L(k)** et **i ≠ k**, alors **s** ← **non**  
Sortir : **s**

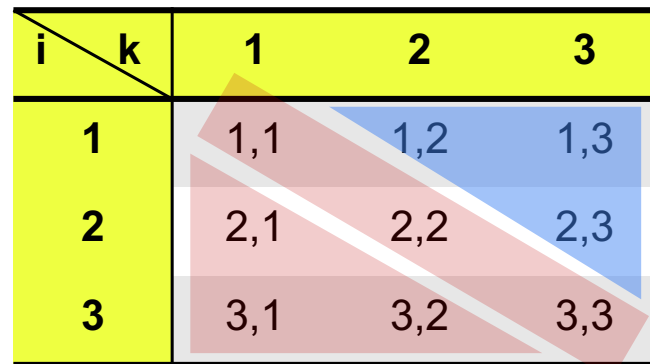
## Tous les 3 différents

**entrée** : liste **L** de 3 objets  
**sortie** : valeur binaire oui/non

**s** ← **oui**  
Si **L(1) = L(2)**, alors **s** ← **non**  
Si **L(1) = L(3)**, alors **s** ← **non**  
Si **L(2) = L(3)**, alors **s** ← **non**  
Sortir : **s**

# Comparaisons dans une boucle imbriquée

i \ k	1	2	3
1	1,1	1,2	1,3
2	2,1	2,2	2,3
3	3,1	3,2	3,3



The diagram illustrates a 3x3 comparison matrix. A diagonal line runs from the top-left to the bottom-right. The upper triangle (above the diagonal) is shaded blue, and the lower triangle (below the diagonal) is shaded red. The diagonal cells are white. The matrix is labeled with 'i' and 'k' in the top-left corner, and the columns are labeled 1, 2, and 3. The rows are labeled 1, 2, and 3. The cells contain pairs of numbers (i,k) representing comparisons.

# Tous différents ?

- **Problème à résoudre :**
  - Parmi une liste de  $n$  objets, identifier si ceux-ci sont tous différents les uns des autres.

## Tous différents

**entrée** : liste  $L$  de  $n$  objets

**sortie** : valeur binaire oui/non

$s \leftarrow \text{oui}$

Pour  $i$  allant de 1 à  $n-1$  :

    Pour  $k$  allant de  $i+1$  à  $n$  :

        Si  $L(i) = L(k)$ , alors  $s \leftarrow \text{non}$

Sortir :  $s$



# Tous différents ?

- **Problème à résoudre :**

- Parmi une liste de  $n$  objets, identifier si ceux-ci sont tous différents les uns des autres.


Tous différents
<b>entrée</b> : liste <b><math>L</math></b> de <b><math>n</math></b> objets <b>sortie</b> : valeur binaire oui/non
<b><math>s \leftarrow \text{oui}</math></b> <b>Pour</b> <b><math>i</math></b> allant de 1 à <b><math>n-1</math></b> : <b>Pour</b> <b><math>k</math></b> allant de <b><math>i+1</math></b> à <b><math>n</math></b> : <b>Si</b> <b><math>L(i) = L(k)</math></b> , alors <b><math>s \leftarrow \text{non}</math></b> <b>Sortir</b> : <b><math>s</math></b>

$i = 1:$	$k = 2, 3, 4, \dots, n$	$n-1$ comp.
$i = 2:$	$k = 3, 4, \dots, n$	$n-2$ comp.
$i = 3:$	$k = 4, \dots, n$	$n-3$ comp.
$\vdots$	$\vdots$	$\vdots$
$i = n-2:$	$k = n-1, n$	2 comp.
$i = n-1:$	$k = n$	1 comp.

$$1 + 2 + 3 + \dots + (n-2) + (n-1) = \frac{n(n-1)}{2} \text{ comp.}$$

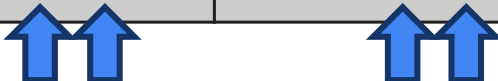
# Algorithme d'Euclide

- L'algorithme d'Euclide utilise une boucle conditionnelle pour trouver le plus grand diviseur commun (pgdc) de deux nombres entiers.

pgdc
<i>entrée</i> : <b>a</b> , <b>b</b> , deux nombres entiers positifs <i>sortie</i> : pgdc(a, b)
<b>tant que</b> <b>b</b> $\neq$ 0 <b>temp</b> $\leftarrow$ <b>b</b> <b>b</b> $\leftarrow$ <b>a</b> mod <b>b</b> <b>a</b> $\leftarrow$ <b>temp</b> <b>Sortir</b> : <b>a</b> 

$\text{pgdc}(a, b) = \text{pgdc}(a-b, b) = \text{pgdc}(a-k.b, b) = \text{pgdc}(a \bmod b, b)$

<b>a = 30</b>	<b>b = 12</b>
$30 = 2 \cdot 3 \cdot 5$	$12 = 2 \cdot 2 \cdot 3$



$\text{pgdc}(30, 12) = 6$

<b>a</b>	<b>b</b>	<b>temp</b>
30	12	12
12	6	6
6	0	



# Résumé Cours 1 – ICC-T

- Programme vs. Algorithme
- Exemples d'algorithmes
- Ingrédients de base : donnés et instructions
- Boucle imbriquée / conditionnelle

[rafael.pires@epfl.ch](mailto:rafael.pires@epfl.ch)

**EPFL**



**Merci**