Information, Calcul et Communication

CS-119(k) ICC – Théorie
Semaine 1

Rafael Pires
rafael.pires@epfl.ch

Lausanne, 20.02.2026

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

Données

Calcul

Traitement

Communication

Transfert

ICC

Information

Données

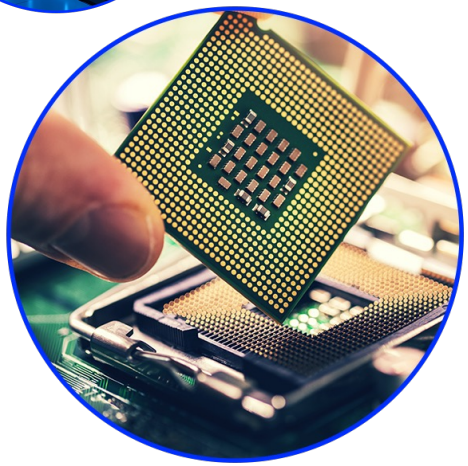
Calcul

Traitement

Communication

Transfert

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.

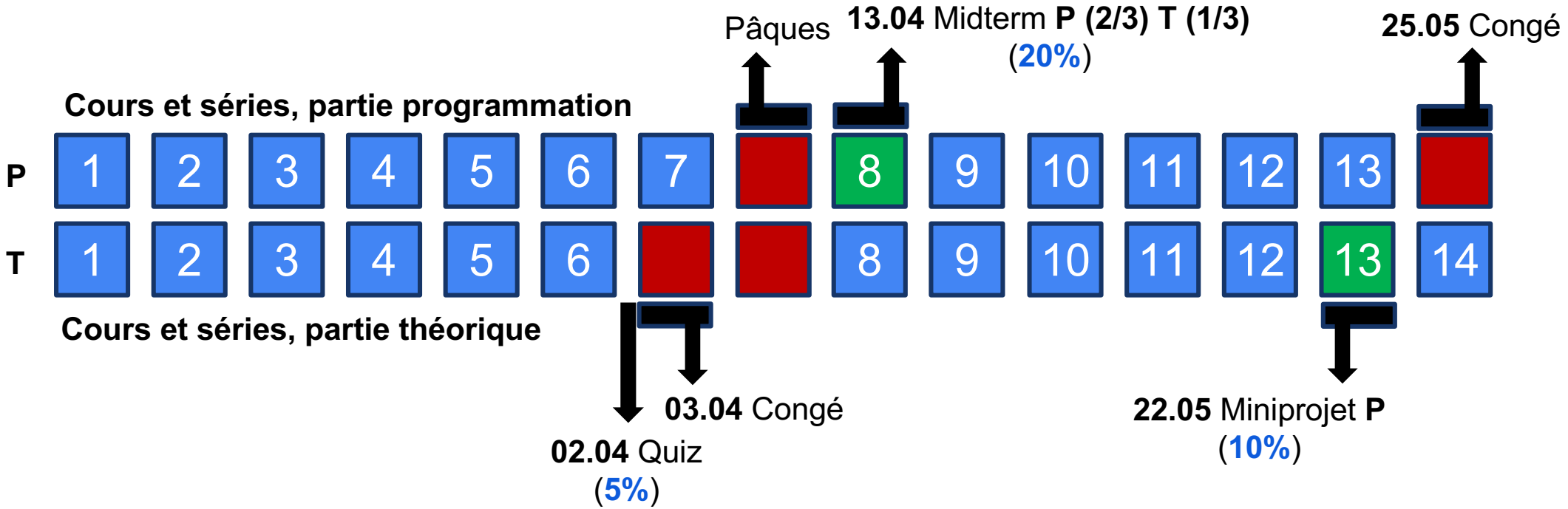
Exercices :

- 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 et MOOC sur courseware.epfl.ch

Programme du cours



??.?? Examen **P (1/3)** et **T (2/3)** (**65%**)
Session d'examen entre 15.06 et 04.07

Programme du cours

Cours et séries, partie programmation

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

Cours et séries, partie théorique

Calcul

Information

Communication

Programme du cours

Calcul

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

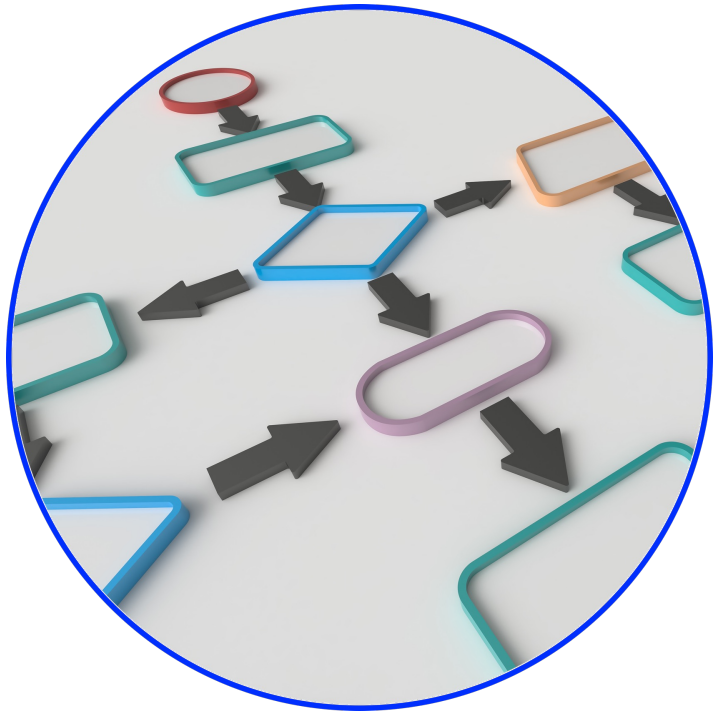
Information

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

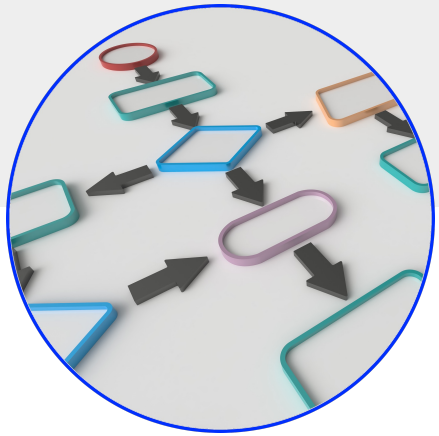
Communication

- Réseau
- Cryptographie

Programme vs. algorithme

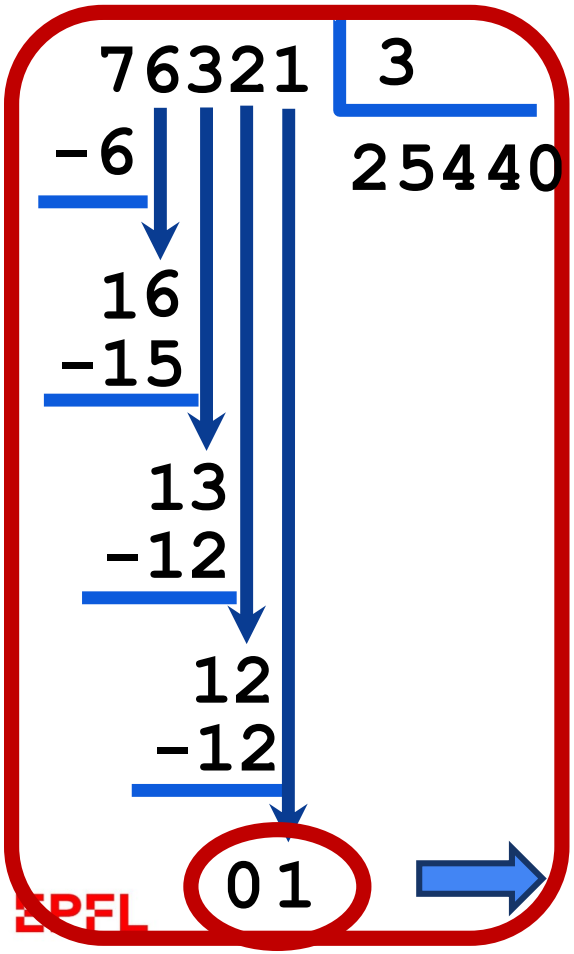


Qu’est-ce qu’un algorithme ?



- 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



$76321 \bmod 3 = 1$

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 = 4.10 + 7 = 4(1+~~2~~) + 7

4 + 7 = 11 = 1(1+~~2~~) + 1

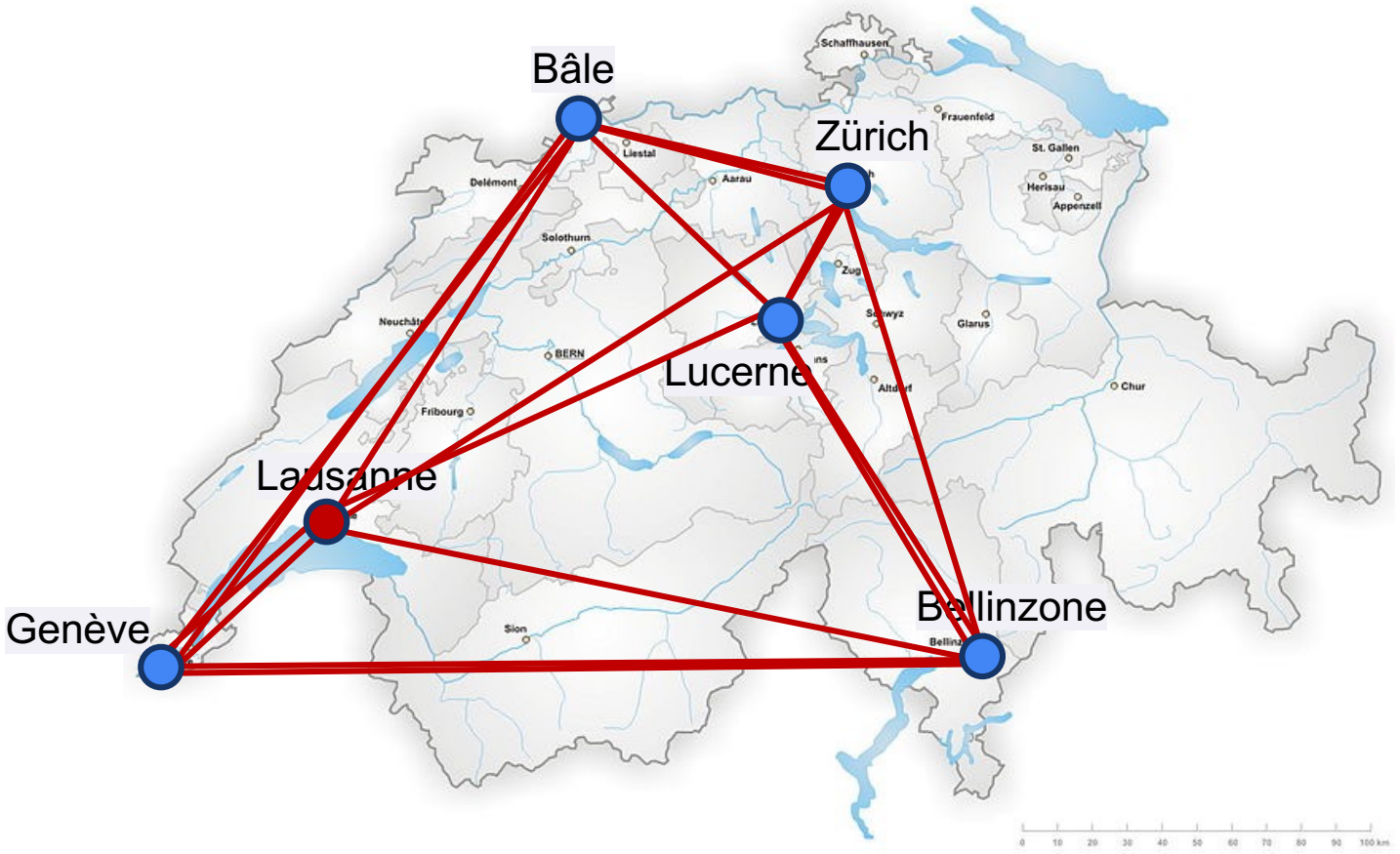
2 ⇒ $47 \bmod 3 = 2$

Exemple 2 : recherche du minimum dans une liste

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

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`**'

Exemple 3 : problème du voyageur de commerce

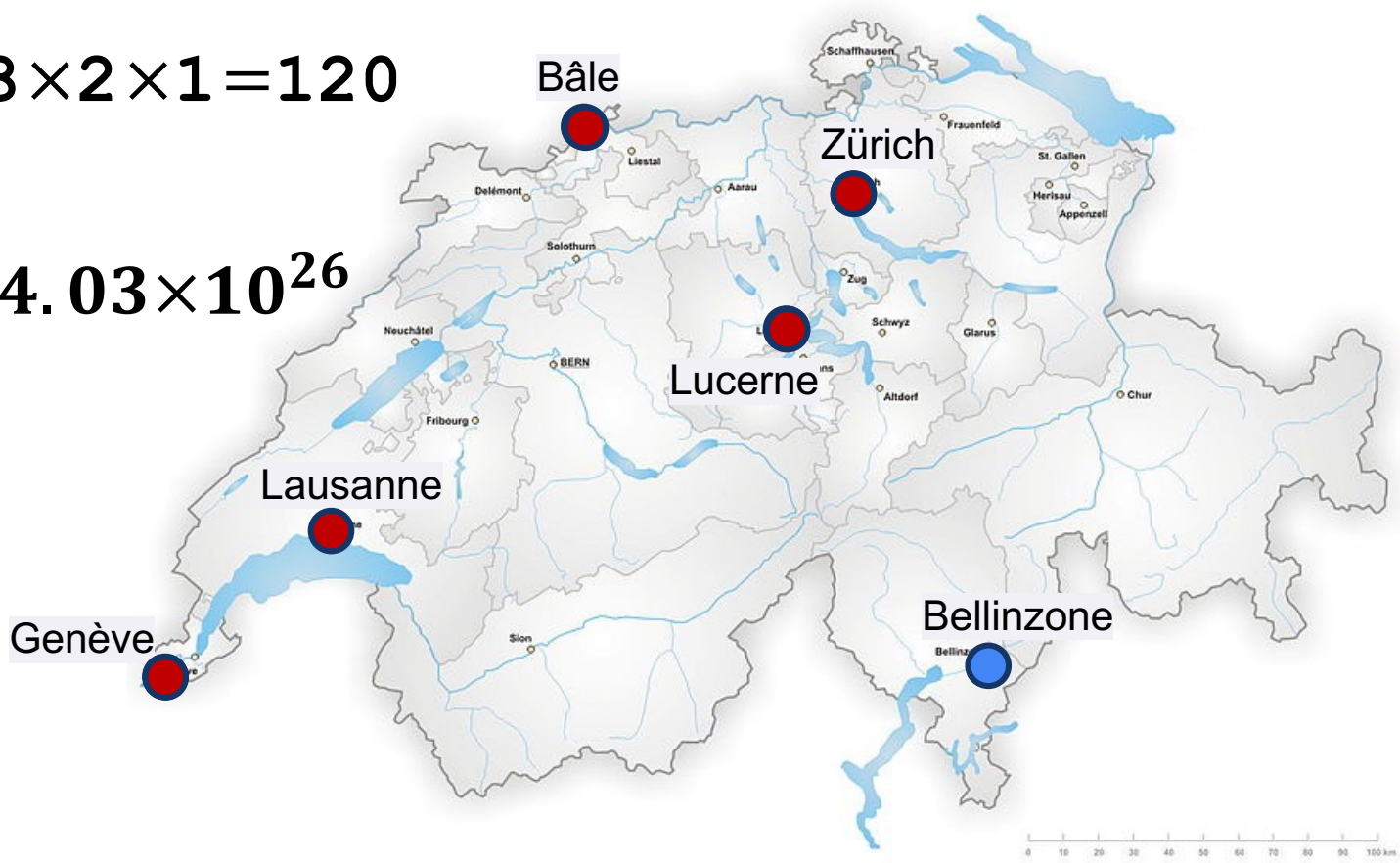


Exemple 3 : problème du voyageur de commerce

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

$$n!$$

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



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

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

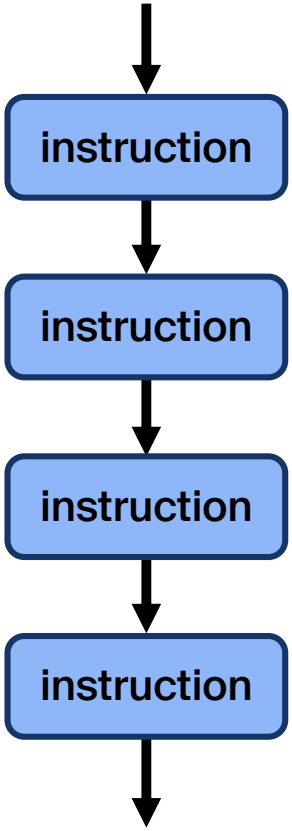
x←1
delta ← $b^2 - 4ac$

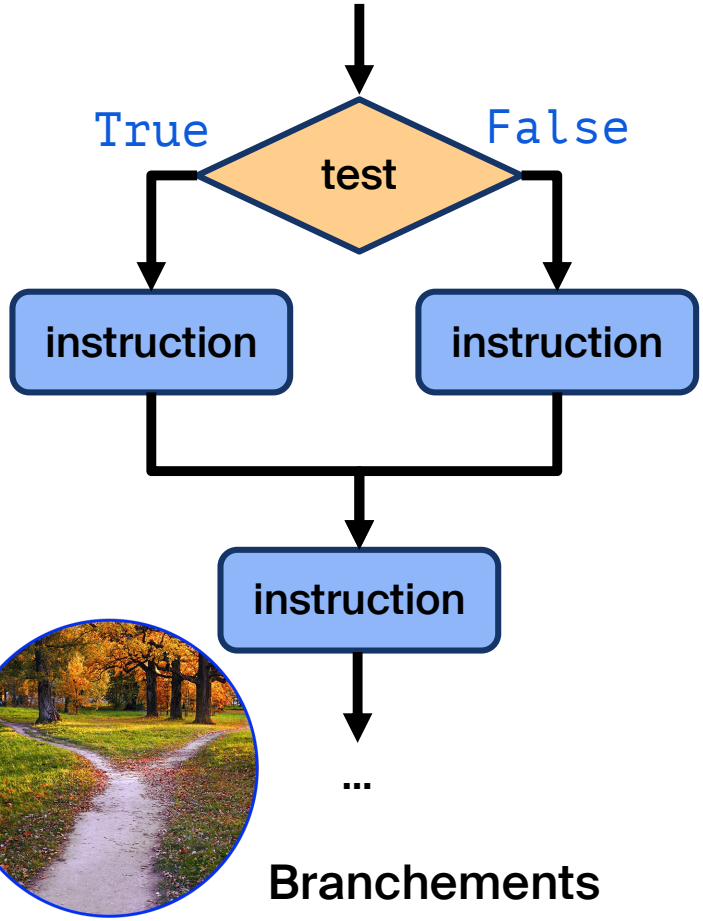
si *delta* < 0, alors …
sinon …

pour i allant de 1 à n, (répeter …)

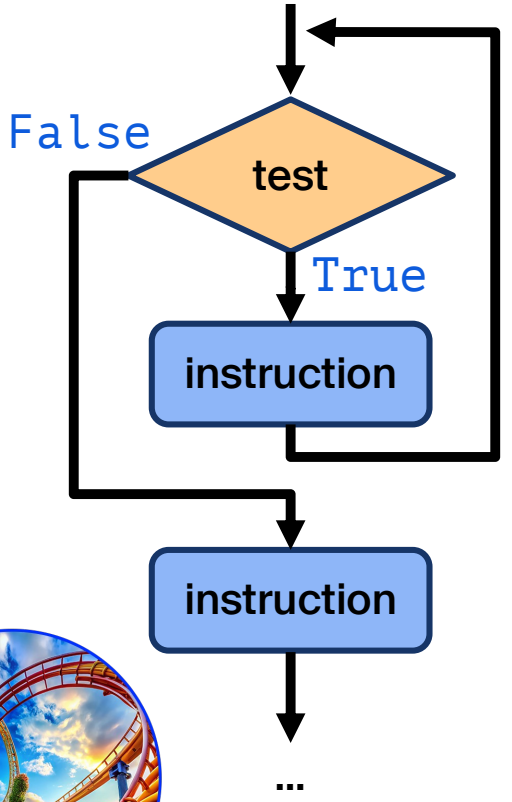
tant que $i \leq 10$, (répeter …)

Structures de contrôle





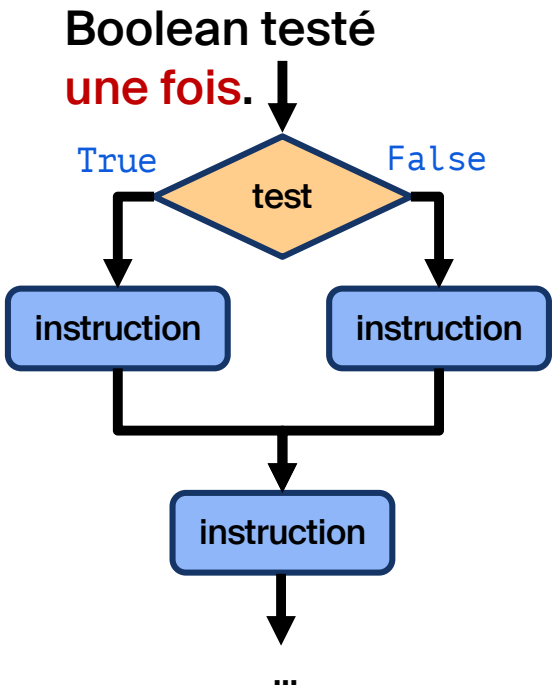
Branchements

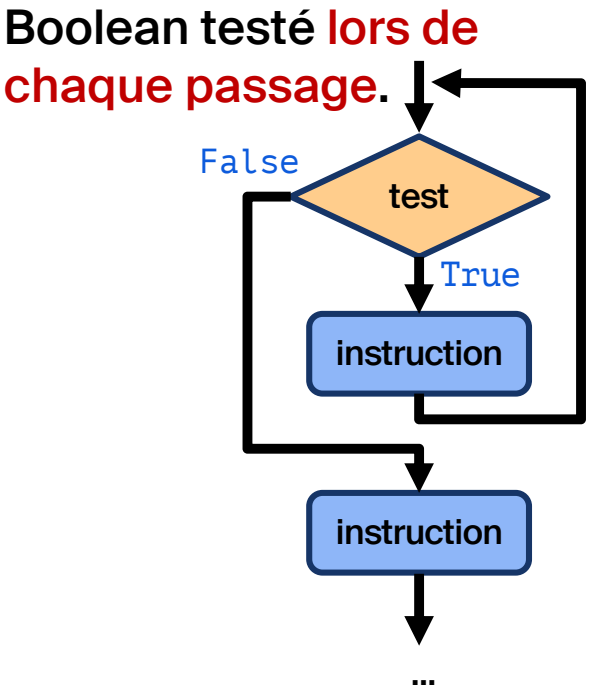


Boucles

Le test

- **Chaque condition nécessite une valeur booléenne pour orienter la suite du traitement**
- Soit vrai (`True`), soit faux (`False`)





Ingrédients de base

Données

- Entrées
- Sorties
- Variables internes

Instructions

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

Entrées

L = (13, 47, 18, 15, 11, 19, 46, 18, 15)
n = 9

Variables internes

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`**'

Branchement conditionnel

Boucle conditionnel

Affectations

Sortie

Pseudo-code

L = (13, 47, 18, 15, 11, 19, 46, 18, 15)
n = 9

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`**'

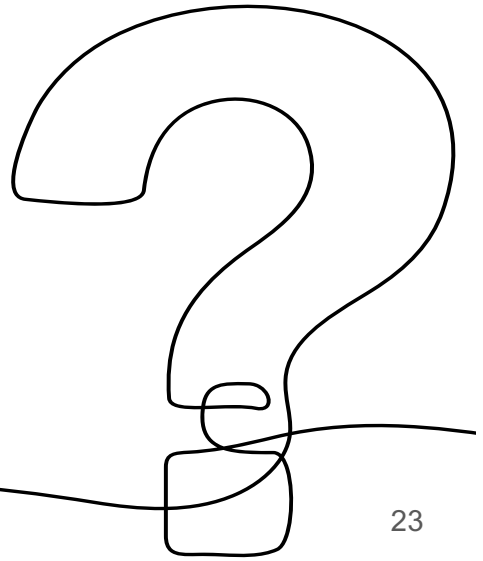
Valeur minimale

entrée : liste ***L*** de nombres entiers, de taile ***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
```

Question : Écrivons un algorithme

Proposer un algorithme permettant de résoudre une équation du second degré $ax^2 + bx + c = 0$ et de déterminer ses solutions réelles.



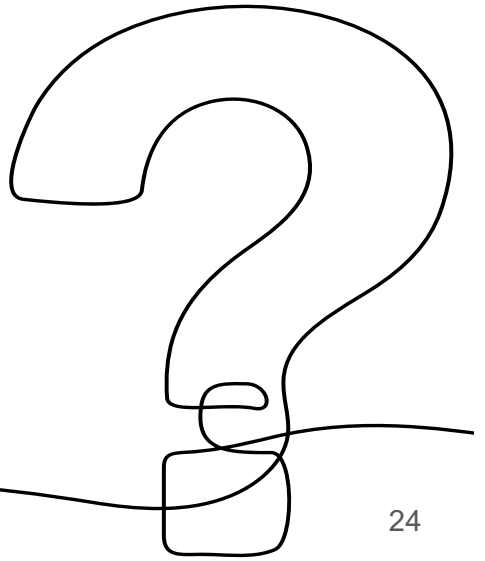
Question : Corrigeons un algorithme

Repérez l'erreur logique dans cet algorithme d'attribution de note et proposez une version corrigée.

Attribution de la note

entrée : *moyenne* m
sortie : *lettre* L

Si $m \geq 16$
  $L \longleftarrow$ "A"
Si $m \geq 12$
  $L \longleftarrow$ "B"
Si $m \geq 8$
  $L \longleftarrow$ "C"
Sortir : L



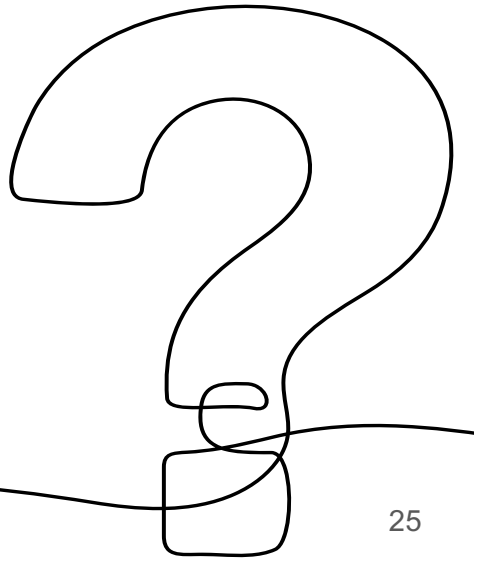
Question : Comprenons un algorithme

On considère l'algorithme ci-dessous, que calcule-t-il ?

Algorithme mystère

entrée : *matrice non vide* M *de taille* $n \times m$
sortie : *valeur* x

$x \longleftarrow M[1,1]$
Pour i *allant de* 1 *à* n
  Pour j *allant de* 1 *à* m
    Si $M[i,j] < x$
      $x \longleftarrow M[i,j]$
Sortir : x

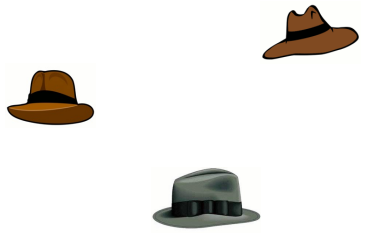


Comparaison tous contre tous

- **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***

9 comparaisons

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***

3 comparaisons

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

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 ← ***oui***
Pour ***i*** *allant de 1 à* ***n****-1 :*
    Pour ***k*** *allant de* ***i****+1 à* ***n*** **:**
        Si ***L***(***i***) = ***L***(***k***), alors ***s*** ← ***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

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

s ← ***oui***
Pour ***i*** *allant de 1 à* ***n****-1 :*
    Pour ***k*** *allant de* ***i****+1 à* ***n*** **:**
        Si ***L***(***i***) = ***L***(***k***), alors ***s*** ← ***non***
Sortir : ***s***

i = 1:	k = 2, 3, 4, …, n	n-1 comp.
i = 2:	k = 3, 4, …, n	n-2 comp.
i = 3:	k = 4, …, n	n-3 comp.
⋮	⋮	⋮
i = n-2:	k = n-1, n	2 comp.
i = n-1:	k = n	1 comp.

$1 + 2 + 3 + \ldots + (n-2) + (n-1) = \frac{n(n-1)}{2}$ 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

entrée : ***a***, ***b***, deux nombres entiers positifs
sortie : pgdc(a, b)

tant que ***b*** ≠ 0
    temp ← ***b***
    b ← ***a*** mod ***b***
    a ← ***temp***
Sortir : ***a***

pgdc(a, b) = pgdc(a-b, b) = pgdc(a-k.b, b) = pgdc(a mod b, b)

a = 30	**b = 12**
30 = 2 . 3 . 5	12 = 2 . 2 . 3

pgdc(30, 12) = 6

a	**b**	**temp**
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

Merci