# Information, Calcul, Communication

## Introduction générale du cours

Sébastien Doeraene /duˈran/

sur base des transparents de J.-C. Chappelier et J. Sam

# Objectifs du cours ICC

- Présenter l'informatique en tant que discipline scientifique
- Exposer ses principes fondamentaux
- Développer la pensée algorithmique (computational thinking)
- Expliquer les bases de fonctionnement du monde numérique
- Sensibiliser à la sécurité dans ce monde numérique
- Vous apprendre à programmer : connaître les bases et au moins deux langages de programmation (C++ ce semestre ; Python au printemps)
- Savoir comment fonctionne un ordinateur et savoir l'utiliser (sous Linux)

# Moyens

- Moodle : support de cours, exercices, compléments, etc. https://moodle.epfl.ch/course/view.php?id=18525
- MOOC (Coursera) pour la programmation : vidéos, quiz, exercices, devoirs corrigés https://www.coursera.org/learn/initiation-programmation-cpp
- Interactions
  - Programmation : exercices jeudi 10h15-12h00 ; restructuration le jeudi suivant 9h15-10h00
  - Théorie : cours vendredi 13h15-15h00 ; exercices 15h15-16h00
- Forums : sur Coursera et sur Moodle

# Organisation du travail (semaine typique)

- Avant le jeudi matin : voir les vidéos du MOOC
- Jeudi 9h15-10h00 : programmation, cours de restructuration
- Jeudi 10h15-12h00 : programmation, exercices encadrés
- Vendredi 13h15-15h00 : théorie, cours ex cathedra
- Vendredi 15h15-16h00 : théorie, exercices
- Après le cours et jusqu'au mercredi suivant : plus d'exercices, puis faire et soumettres les "exercices de programmation" du MOOC (corrigés automatiquement)

# Organisation du travail (semestre)

| | | MOOC | décalage / MOOC | cours prog. | exercices prog. | cours théorie | | exercices théorie | |
|---|---|---|---|---|---|---|---|---|---|
| | | | | 45 min. | 1h45 | 2x45 min. | | 45 min. | |
| | | | | Jeudi 9-10 | Jeudi 10-12 | Vendredi 13-14 | Vendredi 14-15 | Vendredi 15-16 | |
| 1 | 11.09.25 | 1. variables | 0 | Bienvenue/Introduction | variable/expressions | Introduction + Algo 1 | | Algo 1 | 12.09.25 |
| 2 | 18.09.25 | 2. if | 0 | variables / expressions | if – switch | Algorithmes 1 (suite) | | Algo 1 suite | 19.09.25 |
| 3 | 25.09.25 | 3. for/while | 0 | if – switch | for / while | Algo 1 | Algo 2 (stratégies) | Algo 2 | 26.09.25 |
| 4 | 02.10.25 | 4. fonctions | 0 | for / while | fonctions (1) | Algo 2 (stratégies) | Calculabilité | Calculabilité | 03.10.25 |
| 5 | 09.10.25 | | 1 | fonctions (1) | fonctions (2) | Calculabilité | Représentations numériques | Représentations numériques | 10.10.25 |
| 6 | 16.10.25 | 5. tableaux (vector) | 1 | fonctions (2) | vector | Représentations numériques | Signaux + Filtrage | Révisions | 17.10.25 |
| - | 23.10.25 | | | | | | | | |
| 7 | 30.10.25 | 6. string + struct | 1 | vector | array / string | Examen 1 (2h45) | | | 31.10.25 |
| 8 | 06.11.25 | | 2 | array / string | structures | Correction de l'examen | Th. d'échantillonnage | Signaux–Echantillonnage | 07.11.25 |
| 9 | 13.11.25 | 7. pointeurs | 2 | structures | pointeurs | Signaux–Echantillonnage | Compression 1 | Compression 1 | 14.11.25 |
| 10 | 20.11.25 | | - | pointeurs | entrées/sorties | Compression 1 | Compression 2 | Compression 2 | 21.11.25 |
| 11 | 27.11.25 | | - | entrées/sorties | erreurs / exceptions | Compression 2 | Architecture des ordinateurs | Architecture des ordinateurs | 28.11.25 |
| 12 | 04.12.25 | | - | erreurs / exceptions | révisions | Architecture des ordinateurs | Stockage/Réseaux | Stockage/Réseaux | 05.12.25 |
| 13 | 11.12.25 | 8. étude de cas | - | théorie : sécurité | étude de cas | Stockage/Réseaux | théorie : sécurité | Sécurité + Révisions | 12.12.25 |
| 14 | 18.12.25 | | - | Révisions | révisions | Examen final (2h45) | | | 19.12.25 |
| | | | | (ne sont pas sur le MOOC) | (révisions) | (second partie du cours) | (première partie du cours) | | |

# Interactions avec l'enseignant, les assistantes et les assistants

- Durant les séances d'exercices
  - Moyen le plus direct et le plus efficace
- Sur les forums du cours
  - Forum Coursera : questions sur le contenu du MOOC
  - Forum Ed Discussion (Moodle) : questions sur le reste du cours
  - Moyen idéal pour diffuser la connaissance
  - Lieu d'entraide entre vous

Les contacts personnels avec l'enseignant (tels que par e-mail)
devront être strictement réservés aux cas individuels et/ou urgents !

# Livres (1/3)

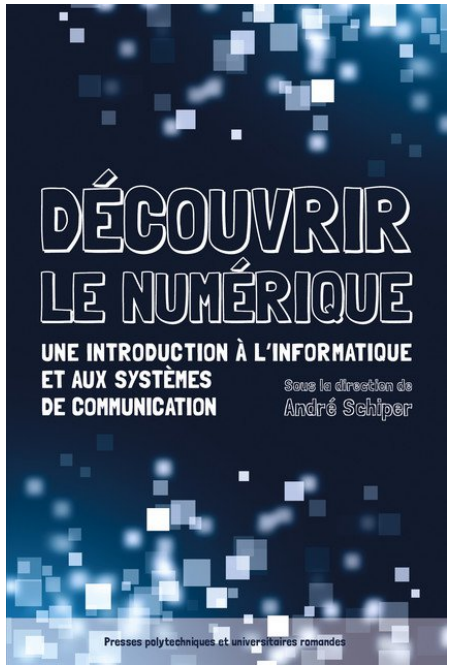
Les éléments fournis (sur le MOOC et sur Moodle) devraient constituer une documentation suffisante pour ce cours.

Si vous souhaitez des livres, les livres suivants sont recommandés.

Pour la théorie :

A. Schiper (éditeur)
*Découvrir le numérique*, PPUR, 2ème édition, 2018

(la première édition est aussi utilisable)




Disponible pour un prix avoisinant les 35 CHF.

Version électronique également disponible.

# Livres (2/3)

Les éléments fournis (sur le MOOC et sur Moodle) devraient constituer une documentation suffisante pour ce cours.

Si vous souhaitez des livres, les livres suivants sont recommandés.

Pour la programmation :

J.-C. Chappelier, J. Sam et V. Lepetit
*Initiation à la programmation en C++*, PPUR, 2ème édition, 2016


eBook téléchargeable gratuitement aux PPUR.

# Livres (3/3)

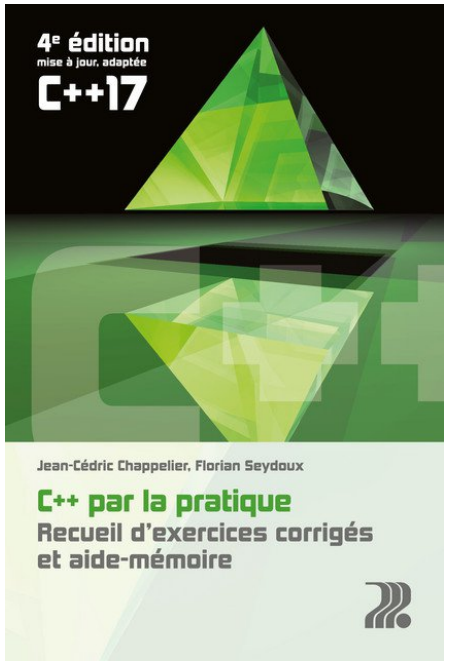
Les éléments fournis (sur le MOOC et sur Moodle) devraient constituer une documentation suffisante pour ce cours.

Si vous souhaitez des livres, les livres suivants sont recommandés.

Pour la programmation :

J.-C. Chappelier, Florian Seydoux
*C++ par la pratique – recueil d'exercices corrigés et aide-mémoire*, PPUR, 4ème édition, 2018

(les éditions précédentes sont aussi utilisables)




Disponible pour un prix avoisinant les 35 CHF.

Version électronique également disponible.

# Évaluations

4 évaluations pendant le semestre :

- Quiz de sécurité (5 %) : à terminer avant le vendredi 3 octobre, 23h58
- Examen 1 (20 %) : vendredi 31 octobre (à confirmer), 13h15-16h00
- Mini-projet (10 %) : du vendredi 14 novembre au vendredi 28 novembre, 23h58
- Examen 2 (55 %) : vendredi 19 décembre (à confirmer), 13h15-16h00

Format des deux examens : 2h45, sur papier — tous documents imprimés autorisés (mais pas de support électronique)

Contenu :

- Examen 1 : théorie module I + programmation jusqu'aux "vectors"
- Examen 2 : tout le cours (théorie et programmation)

# Sensibilisation à la cybersécurité

- L'École tient à sensibiliser aux cyber-risques toutes les personnes membres de la communauté EPFL, notamment pour protéger l'institution et son campus des cyber-attaques.
- Ceci inclut les étudiantes et étudiants

Moyens :

- 13 petites vidéos (43 minutes en tout)
  Lien disponible sur Moodle (presque tout en bas, sous la section "Sensibilisation à la cyber-sécurité")
- Un quiz à passer avant le vendredi 3 octobre 23h58, sur Moodle
  - 2 tentatives
  - 5% de la note finale

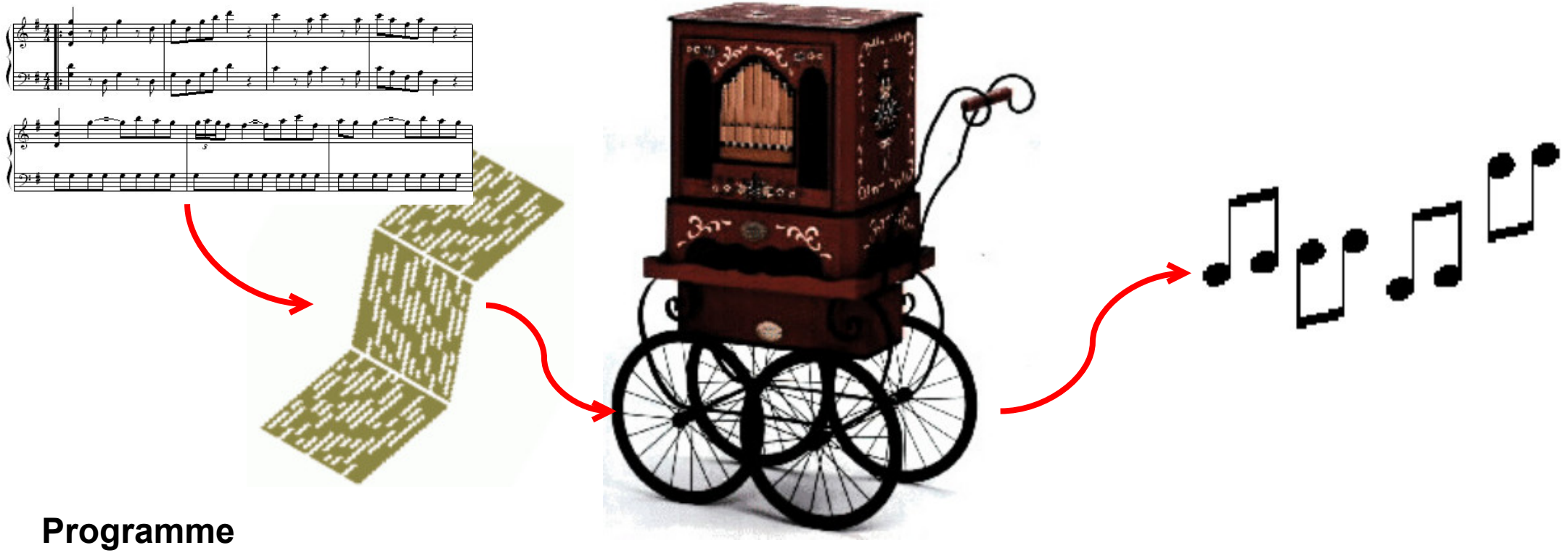
# “Exercices de programmation” du MOOC

- Soumissions obligatoires
- Corrigés automatiquement
- Ne comptent pas pour votre note d’ICC
- Excellents exercices pour pratiquer la programmation
  Une partie essentielle de votre apprentissage

# Qu’est-ce que la programmation ?

- Objectif : permettre l’automatisation d’un certain nombre de tâches à l’aide d’ordinateurs
- Un ordinateur est un exemple d’automate programmable
- Un automate est un dispositif capable d’assurer, sans intervention humaine, un enchaînement d’opérations correspondant à la réalisation d’une tâche donnée.
  Exemples : montres, ramasse-quilles
- Un automate est programmable lorsque la nature de la tâche qu’il est capable de réaliser peut être modifiée volonté.  Dans ce cas, la description de la tâche à réaliser se fait par le biais d’un programme : une séquence d’instructions et de données susceptibles d’être traitées (c.-à-d. « comprises » et « exécutées ») par l’automate.
  Exemples : le métier à tisser Jacquard, l’orgue de barbarie, et l’ordinateur !

# Exemple d'automate programmable



**Programme**

Conception : quelles notes enchaîner ?

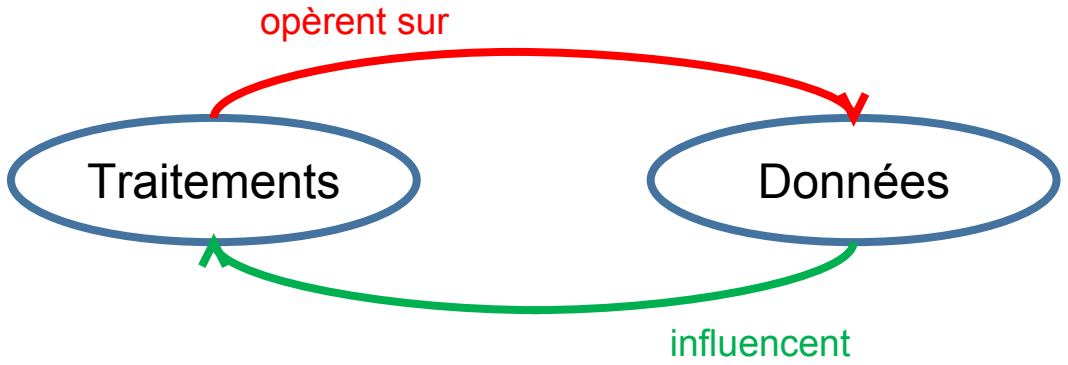
Réalisation : percer les trous aux bons endroits

Exécution : tourner la manivelle

Résultat : mélodie

# La programmation en résumé

Décomposer la tâche à automatiser sous la forme d'une séquence d'intructions (traitements) et de données adaptées à l'automate utilisé.




Formalisation des traitements : algorithmes

► distinguer formellement les bons traitements des mauvais

Formalisation des données : structures de données abstraites

► distinguer formellement les bonnes structures de données des mauvaises

# Les instructions de l'ordinateur

Quelles sont les instructions et les données adaptées à l'ordinateur ?

Ordinateur ~=

- Microprocesseur
  détermine l'ensemble des instructions élémentaires que l'ordinateur est capable d'exécuter
- Mémoire centrale
  détermine l'espace dans lequel des données peuvent stockées en cours de traitement
- Périphériques
  permettent l'échange ou la sauvegarde à long terme des données

Architecture de Von Neumann (1955)

# Les instructions de l'ordinateur (2)

Quelles sont les instructions et les données adaptées à l'ordinateur ?

Ordinateur ~=

- Microprocesseur
- Mémoire centrale
- Périphériques

C'est donc le microprocesseur qui détermine le jeu d'instructions (et le type de données) à utiliser.

On les appelle "Instructions machine", "Langage machine", parfois "Assembleur".

On peut programmer directement l'ordinateur en langage machine ou assembleur, mais c'est fastidieux. De plus, chaque processeur à son jeu d'instructions spécialisé.

# Exemple d'instructions machine

| Programme en Assembleur | | | Signification |
|---|---|---|---|
| 1: LOAD | 10 | 5 | Mettre 5 dans la mémoire 10 |
| 2: CMP | 10 | 0 | Comparer le contenu de la mémoire 10 à 0 |
| 3: JUMP_IF_EQ | +3 | | Si c'est égal, sauter 3 instructions plus loin (à 6) |
| 4: DECR | 10 | | Décrémenter la mémoire 10 |
| 5: JUMP | -3 | | Sauter 3 instructions en arrière (à 2) |
| 6: END | | | |

| Instruction | Code machine | Donnée | Code machine |
|---|---|---|---|
| CMP | 00000000 | -3 | 11111101 |
| DECR | 00000001 | 0 | 00000000 |
| END | 00000010 | 2 | 00000010 |
| JUMP | 00000011 | 3 | 00000011 |
| JUMP_IF_EQ | 00000100 | 5 | 00000101 |
| LOAD | 00000101 | 6 | 00000110 |
| | | 10 | 00001010 |

Le programme ci-dessus correspond donc physiquement en machine à la séquence :
000001000000101000000101000000000000101000000000000010000000011
000000010000101000000011111111010000010

# La notion de langage de programmation

Les instructions machine sont trop élémentaires pour que les humains puissent les utiliser efficacement.

On fournit donc aux programmeuses et programmeurs des instructions de plus haut niveau, plus proches de notre manière de penser et de conceptualiser les problèmes.

Exemples :

| C++ | Python |
| --- | --- |
| `for (int i = 5; i != 0; i--) {`<br>`  cout << i << endl;`<br>`}` | `for i in range(5, 0, -1):`<br>`  print(i)` |

# Langage de programmation (2)

Comment rendre les instructions plus sophistiquées compréhensibles par l'ordinateur ?

- Traduire les séquences d'instructions de haut niveau en instructions machine, exécutables par le microprocesseur

La traduction est effectuée automatiquement, par un autre programme. On appelle un tel programme un compilateur ou interpréteur, en fonction de ses caractéristiques.

L'ensemble des instructions de plus haut niveau qu'un compilateur ou interpréteur est capable de traiter constitue un langage de programmation.

# Interpréteur ou compilateur ?

- En théorie :
  - un interpréteur traduit un programme instruction par instruction
  - un compilateur traduit un programme entier en une fois.
- De nos jours, il n'y a pratiquement plus de différence fondamentale entre un interpréteur et un compilateur.
  - La majorité des langages de programmation existent sur un spectre entre interprétation et compilation.
- Dans le "langage courant" :
  - On parle de compilateur si on utilise manuellement deux phases distinctes (si on applique deux commandes) : la compilation suivie de l'exécution
  - On parle d'interpréteur si on applique qu'une seule commande : l'exécution (mais qui fait de la compilation en interne)

# Compilation et exécution du C++

compilation
commande : g++ hello.cpp -o hello

hello.cpp (source)

```
#include <iostream>

using namespace std;

int main() {
  cout << "Hello World!" << endl;
  return 0;
}
```

hello (exécutable)

```
010100001010101
001010101001110
101111001010001
...
```

exécution
commande : ./hello

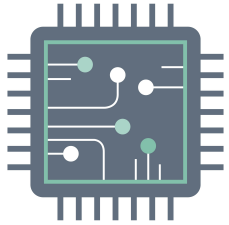


Image by rawpixel.com on Freepik

# Cycle de développement

1. Réfléchir au problème ; concevoir l'algorithme
2. Traduire cette réflexion en un texte C++ (programme source)
3. Traduire ce texte C++ en langage machine (compilation, programme exécutable)
4. Exécution du programme

En pratique :

- Erreurs de compilation
- Erreurs d'exécution
- Résultats incorrects

► Corrections, d'où les cycles

# Ce que j’ai appris aujourd’hui

- Rappels sur l’organisation du cours
  Lire le descriptif du cours sur Moodle (à lire absolument)
- Ce qui caractérise la programmation :
  des traitements et des données
- Ce qu’est et à quoi sert un langage de programmation
- Le cycle de développement

# Quelques considérations sur vos études

# La suite

- Aujourd'hui à 10h15 : exercices de programmation
  - CO 5 et INM 10 si vous souhaitez utiliser votre propre ordinateur (INM 11 si besoin)
  - CO 020-023 si vous souhaitez utiliser un ordinateur de l'école
- Demain à 13h15 : cours sur la théorie en BCH 2201, puis exercices en BS 160 et BS 170
- La semaine prochaine : cours de restructuration sur les variables et expressions
- D'ici là : faites plus d'exercices ; soumettez les "exercices de programmation" sur Coursera

| | | MOOC | décalage / MOOC | cours prog. | exercices prog. | cours théorie | | exercices théorie | |
|---|---|---|---|---|---|---|---|---|---|
| | | | | 45 min. | 1h45 | 2x45 min. | | 45 min. | |
| | | | | Jeudi 9-10 | Jeudi 10-12 | Vendredi 13-14 | Vendredi 14-15 | Vendredi 15-16 | |
| 1 | 11.09.25 | 1. variables | 0 | Bienvenue/Introduction | variable/expressions | Introduction + Algo 1 | | Algo 1 | 12.09.25 |
| 2 | 18.09.25 | 2. if | 0 | variables / expressions | if – switch | Algorithmes 1 (suite) | | Algo 1 suite | 19.09.25 |
| 3 | 25.09.25 | 3. for/while | 0 | if – switch | for / while | Algo 1 | Algo 2 (stratégies) | Algo 2 | 26.09.25 |
| 4 | 02.10.25 | 4. fonctions | 0 | for / while | fonctions (1) | Algo 2 (stratégies) | Calculabilité | Calculabilité | 03.10.25 |
| 5 | 09.10.25 | | 1 | fonctions (1) | fonctions (2) | Calculabilité | Représentations numériques | Représentations numériques | 10.10.25 |
| 6 | 16.10.25 | 5. tableaux (vector) | 1 | fonctions (2) | vector | Représentations numériques | Signaux + Filtrage | Révisions | 17.10.25 |
| - | 23.10.25 | | | | | | | | |
| 7 | 30.10.25 | 6. string + struct | 1 | vector | array / string | Examen 1 (2h45) | | | 31.10.25 |
| 8 | 06.11.25 | | 2 | array / string | structures | Correction de l'examen | Th. d'échantillonnage | Signaux–Echantillonnage | 07.11.25 |
| 9 | 13.11.25 | 7. pointeurs | 2 | structures | pointeurs | Signaux–Echantillonnage | Compression 1 | Compression 1 | 14.11.25 |
| 10 | 20.11.25 | | - | pointeurs | entrées/sorties | Compression 1 | Compression 2 | Compression 2 | 21.11.25 |
| 11 | 27.11.25 | | - | entrées/sorties | erreurs / exceptions | Compression 2 | Architecture des ordinateurs | Architecture des ordinateurs | 28.11.25 |
| 12 | 04.12.25 | | - | erreurs / exceptions | révisions | Architecture des ordinateurs | Stockage/Réseaux | Stockage/Réseaux | 05.12.25 |
| 13 | 11.12.25 | 8. étude de cas | - | théorie : sécurité | étude de cas | Stockage/Réseaux | théorie : sécurité | Sécurité + Révisions | 12.12.25 |
| 14 | 18.12.25 | | - | Révisions | révisions | Examen final (2h45) | | | 19.12.25 |
| | | | | (ne sont pas sur le MOOC) | (révisions) | (second partie du cours) | (première partie du cours) | | |