ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE
EIDGENÖSSISCHE TECHNISCHE HOCHSCHULE – LAUSANNE
POLITECNICO FEDERALE – LOSANNA
SWISS FEDERAL INSTITUTE OF TECHNOLOGY – LAUSANNE

Faculté Informatique et Communication
Cours « Information, Calcul, Communication » SMA (CS-119(l))
S. Doeraene

« Information, Calcul, Communication » CS-119(l)

Présentation générale du cours

IX MMXXV

1 Introduction

Ce document a pour but de vous informer sur la pédagogie du cours « Information, Calcul, Communication » donné à la Section MA, son mode de fonctionnement et divers autres aspects liés à son organisation.

Veuillez lire l'entièreté de ce document, et ne ratez surtout pas **la section 6** !

Table des matières :

2 Un cours pourquoi ? pour qui ?

L'objectif premier de ce cours est de vous faire acquérir
- d'une part les **concepts fondamentaux** de l'Informatique en tant que discipline scientifique, ainsi qu'une certaine « pensée algorithmique » ;
- et d'autre part **une base commune** en programmation, nécessaire pour mener à bien la suite de vos études à l'EPFL.

La partie théorique est organisée en trois modules :
- calcul (algorithmes, récursion, complexité, représentation des nombres) ;
- information (échantillonnage, reconstruction, th. de Nyquist-Shannon, compression, 1er th. de Shannon) ;
- systèmes et sécurité (ordinateur de von Neumann, mémoires et réseaux, menaces et défenses, cryptographie à clef secrète, RSA).

La partie pratique vise à :
- enseigner les notions fondamentales communes à la plupart des langages de programmation généralistes et « impératifs » (variables, expressions, structures de contrôle, fonctions, entrées-sorties, etc.) ;
- les illustrer au moyen du langage C++ ;
- et vous familiariser avec un environnement de développement informatique.

Les notions vues au semestre d'automne seront consolidées au semestre de printemps, notamment par la réalisation d'un projet dans le cours « Développement Logiciel », cette fois en Python (CS-121).

Ce cours est en premier lieu conçu et organisé **pour des personnes débutantes en programmation** et ne demande *aucune connaissance préalable* en informatique. Mais cela ne veut pas dire qu'il ne soit pas *exigeant* ! Son ambition est de faire de vous des personnes éduquées en sciences de l'information et des programmeuses et programmeurs *compétents*.

Pour les notions plus avancées, ce cours a aussi pour objectif de *consolider l'acquis* des étudiantes et étudiants qui auraient déjà programmé au préalable, en y apportant les bases plus formelles qui font parfois défaut dans un apprentissage autodidacte (méthodologie, notions algorithmiques, bonnes pratiques, etc.).

En raison de votre grande hétérogénéité de niveaux en programmation, le principe pédagogique fondamental de ce cours est de donner *à chacune et chacun* les moyens de progresser à *son* niveau.

Ce principe a conduit à introduire plusieurs éléments :
- un **accès diversifié** au contenu : transparents et vidéos du cours, exercices, fiches résumé, livre conseillé et références externes (« en ligne » et bibliographiques) sont les différents supports mis à disposition ;
- un **accès hiérarchisé par niveau** : quatre niveaux d'exercices (voir section 3.3.3).

Différentes indications de niveaux sont données : niveaux des exercices, commentaires oraux de l'enseignant, etc. Sachez en faire bon usage !

En plus de vous enseigner le langage C++ lui-même, ce cours a aussi et surtout pour objectif de vous sensibiliser à l'importance des aspects *méthodologiques* liés à la programmation. Ainsi, il s'intéresse en priorité au **quoi** (les concepts : structures de contrôle, types de données, etc.) et au **pourquoi** (la raison de leur existence). Il va également vous donner des indications sur le **comment** (algorithmique, règles et conseils pour produire de bons programmes, etc.).

C'est là où des personnes ayant déjà programmé auparavant sont souvent *désavantagées*. En dehors de l'université, vous avez probablement programmé « sur le tas », sans cadre formel clair. Ne vous reposez pas sur vos acquis : profitez de votre expérience pour redécouvrir la programmation sous un nouvel angle.

Pour remplir les objectifs multiples de ce cours, les outils pédagogiques suivants sont mis à votre disposition :
- le site du cours sur Moodle :

`https://moodle.epfl.ch/course/view.php?id=18525`

sur lequel vous trouverez un accès aux transparents du cours, aux séries d'exercices et à leurs corrigés ainsi que des références utiles ;

— pour la partie pratique (programmation), un MOOC (Massive Open Online Course) :
`https://www.coursera.org/learn/init-prog-cpp`

avec tous ses outils pédagogiques (voir section 3.1) ;

— des informations générales sur l'environnement de programmation (voir section 5.4) ;

— des forums de discussion : sur le site Ed Discussion (accessible via le site Moodle du cours) et sur celui du MOOC (voir section 5.1).

Ces différents outils sont décrits plus en détail dans la suite de ce document.

3 Sous quelle forme le cours est-il donné ?

Le cours est enseigné en *deux* parties, assez séparées : la partie théorie (les vendredis) et la partie programmation C++ (les jeudis).

La partie programmation C++ est entièrement donnée sous forme de « classe inversée ». L'introduction du contenu de chaque semaine est mise à disposition dans un MOOC (Massive Open Online Course), sous la forme de vidéos à regarder *avant* de venir en classe. La principale activité d'apprentissage consiste ensuite en une séance d'exercices encadrée. Enfin, un cours en auditoire fait un *retour* (ou restructuration) sur le contenu vu précédemment. Si vous attendez de venir au cours pour avoir le contenu nécessaire à réaliser les exercices de la semaine, vous aurez du mal.

Au contraire, arrivez au cours du jeudi matin *avec vos questions*. Ce sont vos questions qui orienteront le contenu des séances du jeudi, contenu qui se retrouvera au menu des examens.

La partie théorie est donnée de manière plus classique. Un cours *ex cathedra* en auditoire de deux heures est suivi d'une heure d'exercices. En début du cours suivant, on reviendra volontiers sur les notions de la semaine écoulée, suivant vos questions.

Entre la séance d'exercices de chaque partie et le cours qui suit, il est fortement conseillé de pratiquer des exercices supplémentaires non encadrés.

3.1 MOOC (Massive Open Online Course) de programmation

Un MOOC de *8 semaines*, « Initiation à la programmation (en C++) [1] », a été créé dans le but d'offrir de façon pédagogique le minimum de bases nécessaires à tout apprentissage de la programmation. Les personnes ayant déjà de l'expérience en C++ en particulier survoleront probablement ce contenu.

Il constitue une *base essentielle* de votre apprentissage pour la partie pratique de ce cours.

Vous y trouverez :

— des vidéos de cours, d'une dizaine de minutes environ, expliquant en détails un point précis ;
ces vidéos sont par ailleurs ponctuées de quiz qui vous permettent de vérifier votre compréhension de ce qui est présenté ;

— des quiz hors vidéo dont le but est de vérifier votre acquisition des concepts présentés dans la vidéo ;
notez que certains de ces quiz sont particulièrement tordus, et ne reflètent pas nécessairement le type de questions des examens ;

— des tutoriels, exercices reprenant des exemples du cours et dont le corrigé est donné au fur et à mesure de la donnée de l'exercice lui-même ;

1. `https://www.coursera.org/learn/init-prog-cpp`.

ils sont conseillés comme un premier exercice sur un sujet que vous ne pensez pas encore assez maîtriser pour aborder par vous-même un exercice « classique » ;
— des exercices, libres (le corrigé est donné), vous permettant de **mettre en pratique** les concepts présentés ;
la pratique *autonome* de ces exercices est une clef fondamentale de votre apprentissage de la programmation ; ne la négligez pas ;
— des « devoirs », exercices à rendre et qui seront corrigés automatiquement.

Pour ce cours, il est impératif que vous rendiez ces devoirs notés (« *Exercices de programmation* »/« *Programming assignment* » dans la terminologie Coursera, la plate-forme offrant notre MOOC). Même s'ils n'entrent pas dans la note finale, ils constituent un excellent entraînement pour les examens que vous aurez ici, sur le site de l'École. Je leur donne donc un caractère *obligatoire* : il est nécessaire que vous les ayez abordés avant les examens respectifs, mais je vous conseille vivement d'être à jour chaque semaine. La programmation s'apprend en programmant.

Je ne demande par contre rien du tout relativement au certificat Coursera : ce certificat est totalement indépendant du présent cours.

Note importante : merci de vous inscrire au MOOC sur Coursera si possible avec votre adresse email EPFL. Si ce n'est pas possible, envoyez-moi un email[2] m'indiquant avec quelle adresse vous vous êtes inscrit-e sur Coursera.

Un dernier conseil pour un cours « inversé » comme celui-ci (c'est-à-dire où il faut regarder le cours en vidéo avant de venir en classe) : agendez-vous un ou des créneaux hebdomadaires fixes pour regarder les vidéos et travailler la matière de votre côté.

3.2 Séances en auditoire

Le créneau de « cours » réservé à l'emploi du temps réunit toutes les étudiantes et étudiants dans un auditoire.[3] On rappelle que, pour la partie programmation, l'heure de cours revient sur la matière de la semaine *écoulée*. Il est donc absolument nécessaire de voir les vidéos du MOOC *AVANT* de venir en cours et aux exercices !

Afin de faciliter la prise de notes et de vous permettre de préparer vos éventuelles questions, les transparents (ainsi que tout le reste du matériel du cours) sont d'ores et déjà disponibles sur Moodle. De petits changements sont possibles sur les transparents jusqu'à la date du cours. Les solutions aux séances d'exercices apparaîtront le lendemain de la première séance les concernant. Des compléments oraux, sous forme d'exemples additionnels ou de discussions, sont souvent apportés au tableau et sur éditeur de code pendant les cours *ex cathedra*.

Les examens portent sur la matière qui est enseignée dans les MOOCs **ET** dans les cours *ex cathedra*. Il est donc important de bien connaître le contenu (et le style d'enseignement) pour arriver à bien se préparer.

3.3 Séances d'exercices et travail à la maison

3.3.1 Organisation générale

Trois heures d'exercices sont prévues par semaine (une pour la partie théorique et deux pour la partie pratique). Cependant, le temps requis pour résoudre les exercices peut varier, parfois considérablement, en

2. `sebastien.doeraene@epfl.ch`
3. Pour les étudiant-e-s ne *pouvant pas* se rendre sur le campus à l'heure du cours, ces séances seront disponibles en vidéo différée immédiatement après les cours ; mais ce moyen ne doit pas être considéré comme le mode normal pour suivre le cours.

fonction de vos connaissances préalables et votre préparation. Il relève donc de votre propre responsabilité de compléter ces séances par la quantité de travail « à la maison » appropriée.

Il est prévu qu'une étudiante moyenne devrait consacrer trois à quatre heures *supplémentaires* de travail personnel par semaine, et un étudiant totalement novice jusqu'à cinq heures. Considérez par ailleurs les heures d'exercices affichées à l'emploi du temps comme des heures d'assistance : nous sommes là pour vous aider. Organisez donc votre travail de sorte à faire chez vous les choses qui vous semblent abordables et réservez les aspects difficiles et surtout les questions pour les séances d'exercices en salle. Pensez aussi à utiliser les forums du cours (voir plus loin).

L'heure d'exercices de la partie théorique (vendredi) se fait uniquement sur papier (avec éventuellement ponctuellement l'aide d'une calculatrice).

Les deux heures d'exercices de la partie pratique (jeudi) se déroulent dans une salle d'ordinateurs (voir section 5.2) où vous pouvez travailler seule ou avec des camarades, sur le matériel de l'école ou sur votre propre ordinateur portable.

Plusieurs assistantes et assistants (ainsi que moi-même) sont présents pendant ces séances pour vous aider. Ils répondront à tous types de questions (sur le cours). Je vous encourage à discuter de vos solutions, vos programmes et de vos problèmes éventuels avec elles et eux. De plus, il est encouragé de discuter de vos solutions entre vous, sans pour autant vous reposer sur les solutions de vos camarades. Les assistantes et assistants ne s'imposent pas, mais attendent que vous les sollicitiez. Sachez profiter pleinement de leur présence !

Pour chaque séance, il y a une série d'exercices à résoudre de manière indépendante. L'énoncé est mis à disposition sur le site Moodle (EPFL) du cours pour la partie théorie et sur le site du MOOC (Coursera) pour la partie programmation. Le corrigé sera également mis à disposition au même endroit. Tout le contenu de la partie programmation est également disponible dans le livre « *C++ par la pratique* » de Jean-Cédric Chappelier et Florian Seydoux (éditions PPUR), en vente à « La Boutique » de l'EPFL. Une version électronique est également disponible.

Remarque importante : Il est vivement recommandé de participer aux séances d'exercices :
- cela vous permet d'assurer la *régularité* de votre progression, qui est une clef essentielle de la réussite de ce cours ;
- cela vous permet de bénéficier de l'aide des assistants et assistantes ;
- et cela nous permet de vous donner un retour sur votre niveau, afin de vous aider à vous évaluer et si nécessaire à vous indiquer comment progresser.

3.3.2 Où trouver les exercices ?

Pour les exercices de la partie théorie (vendredis) : sur la page Moodle du cours, dans la section de la semaine concernée.

Pour les exercices de programmation (jeudis) : il y a principalement deux sources possibles, presque identiques :
- soit le MOOC (sur Coursera) : sous le lien « Lecture : exercices » ;

  vous y trouverez :
  - des tutoriels (parfois aussi appelés « exercice niveau 0 ») ;
  - des exercices « normaux » (niveaux 1 à 2) ;
  - des exercices « avancés » (niveau 3, appelés « supplémentaires facultatifs »), intéressants pour progresser encore plus.

  Vous trouverez aussi au même endroit les corrigés correspondants (texte et codes sources).
- soit dans le livre « *C++ par la pratique* » (avec les corrigés).

Ces deux sources ont une très grosse intersection. Le MOOC a quelques exercices plus simples en plus ici ou là, et a surtout les exercices obligatoires à rendre (point suivant). Le livre a quelques exercices supplémentaires (tirés d'anciens examens).

Point important :

Le site Moodle offre **en plus** quelques exercices de programmation *spécifiques* : ce sont des exercices tout spécialement conçus pour ce cours ICC, pour faire le lien entre la théorie du vendredi et la programmation du jeudi. Je vous les recommande particulièrement (d'ici quelques semaines ; au début du semestre nous n'avons pas encore vu assez de matériel en programmation).

3.3.3 Catégorisation des exercices par niveaux

Le contenu proposé lors des séries d'exercices est volontairement sur-dimensionné : elles contiennent sensiblement plus de matériel qu'il n'est faisable en deux heures, surtout pour les novices. Ceci est conçu afin que chacune et chacun *choisisse*, selon ce qui l'intéresse, ce qu'il souhaite approfondir.

Il ne vous est donc bien entendu pas demandé de tout faire. Les séries sont à voir comme du matériel d'entraînement dans lequel vous pouvez puiser au gré de vos besoins, un peu comme un livre d'exercices (lequel est par ailleurs vivement recommandé).

Comme évoqué en fin de section précédente, nous vous fournissons deux types d'exercices de programmation :

- des exercices généraux, sur le site du MOOC ;
- à partir de la semaine 6, des exercices *spécifiques* à ce cours, qui font le lien entre la partie théorique du cours et la programmation ; ils sont disséminés au fil des semaines.

À partir de la semaine 6, je vous encourage donc à commencer par quelques exercices généraux, puis à visiter ceux spécifiques à ce cours. Je pense que ces derniers vous permettront de progresser sur les deux parties du cours.

Dans le but de vous aider à vous organiser, les exercices sont organisés par niveaux de difficulté (de 0 à 3) :

- **niveau 0** (ou « tutoriels ») : reprise pas à pas d'un exemple du cours ; ils peuvent sans problème être sautés par celles et ceux qui estiment avoir une suffisamment bonne compréhension de la programmation de base et du cours du jour ;

- **niveau 1** : ces exercices élémentaires devraient pouvoir être faits par toutes et tous dans un temps raisonnable (30 à 45 minutes maximum au début, 20 min. en « régime de croisière », 10 min. max. avec plus d'expérience) ; ils permettent de travailler les bases ;

- **niveau 2** : ces exercices plus avancés devraient être abordés par toutes et tous, sans forcément être finis au début ; la reprise de l'exercice avec la correction devrait constituer un bon moyen de progresser ;

  c'est par ailleurs le niveau que je considère que vous devrez avoir acquis à la fin du cours ; c'est donc à ce niveau que je fixe le 4.0 des examens ;

  c'est aussi à ce niveau que sont les « exercices de programmation » du MOOC (exercices obligatoires, à soumettre) ;

- **niveau 3** (ou « exercices supplémentaires » dans le MOOC) : ces exercices d'un niveau *avancé* sont pour les plus motivées et habiles d'entre vous ; ils peuvent *dans un premier temps* être ignorés, mais doivent être repris, si nécessaire avec la correction, lors des révisions afin de progresser ;

  les techniques qu'ils utilisent, leur niveau de difficulté, peuvent *ponctuellement* être présents en examen.

Notez que les niveaux sont déterminés en fonction du moment où la série d'exercices est offerte. Il est clair qu'un même exercice donné plus tard au cours de l'année (par exemple au moment de l'examen de fin de semestre) serait considéré comme plus facile !

Je considère que le travail *minimum* par semaine consiste en **deux exercices de niveau 1 et un de**

niveau 2.

Bien entendu, votre niveau en programmation ne peut qu'être amélioré si vous réalisez d'avantage d'exercices. Il est donc conseillé, comme déjà évoqué plus haut, de compléter le travail réalisé pendant les séances d'exercices en salle, par du travail personnel hors des heures imparties à ce cours.

Les séries d'exercices hebdomadaires ne sont pas notées et il ne vous est pas demandé de les rendre (sauf les « *Exercices de programmation* »/« *Programming assignment* » dans la terminologie Coursera, qui sont à soumettre impérativement). Elles sont à considérer comme une aide à l'apprentissage et comme une préparation aux examens. Si vous n'arrivez pas à terminer une série d'exercices pendant la semaine, il est possible d'en consulter le corrigé et d'en étudier les détails. Essayez cependant de *résoudre un maximum de problèmes par vous-même*. C'est le meilleur moyen d'apprendre. Si, par contre, il vous reste du temps, complétez la série par un peu de curiosité et d'expérimentation personnelle : ajoutez à votre gré d'autres fonctionnalités à vos programmes, consultez la documentation, etc. En règle générale, toute manipulation sérieuse sur l'ordinateur augmentera vos connaissances.

Pour quelques unes et quelques uns d'entre vous, mêmes les exercices de niveau 3 vous paraîtront très simples. Tant mieux pour vous ! Si vous voulez corser la chose : tentez de les résoudre entièrement *sans les exécuter* (voire sans les compiler !). Ne vérifiez votre solution que lorsque vous la pensez finie. Lors des examens, vous devrez écrire de petits programmes sur papier ; c'est l'occasion de tester si vous pouvez réellement *prédire* le comportement de vos programmes.

4 Comment le cours est-il évalué

Ce cours est une « branche de semestre » comptant coefficient 6 dans le Bloc 2.

Les connaissances que vous aurez acquises seront évaluées à l'aide de deux examens (sur papier uniquement) et un « mini projet » de programmation (à faire à la maison) :

- un examen de 2h45 le **vendredi 31 octobre (à confirmer) de 13h15 à 16h00**, portant sur le module I de la partie théorique du cours et sur la partie de programmation C++ couverte jusque là (vectors) ; cet examen est « avec document » (tout document papier autorisé, mais aucun matériel électronique) ;
- un examen final de 2h45, le **vendredi 19 décembre (à confirmer) de 13h15 à 16h00**, portant sur l'*entièreté* du cours, parties théorique et pratique ; cet examen est « avec document » ;
- le « mini projet » sera un exercice de programmation à faire chez soi de façon autonome, **du 14 au 28 novembre**.

La note finale du cours est calculée suivant la répartition :

- quiz de sécurité appliquée : 5% ;
- examen 1 : 30% ;
- mini projet : 10% ;
- examen final : 55%.

Plus précisément : soit p_x le nombre de points obtenus à l'épreuve x (0 en cas d'absence) sur un total maximal pour cette épreuve de t_x ; la note publiée pour cette épreuve est alors l'arrondi suivant :

$$n_x = 1 + 0.25 \cdot \left\lceil 20 \cdot \frac{p_x}{t_x} \right\rceil$$

La note finale N est calculée, en fin de semestre, directement sur les points obtenus (et non pas sur les notes intermédiaires) par

$$N = 1 + 0.25 \cdot \left\lceil 20 \cdot \sum_x \theta_x \frac{p_x}{t_x} \right\rceil$$

où θ_x est le coefficient de l'épreuve x (par exemple 0.55 pour l'examen final). La note finale peut donc être *plus basse* que la moyenne pondérée des notes intermédiaires (si vous avez beaucoup bénéficié des arrondis vers le haut sur chaque note intermédiaire).

Enfin, le rendu de tous les « exercices de programmation » du MOOC est obligatoire avant le dernier examen. Je vous conseille vivement d'être déjà à jour au fil de chaque semaine, et certainement avant le premier examen. Il s'agit d'un entraînement essentiel.

Je ne demande par contre rien du tout relativement au certificat Coursera : ce certificat est totalement indépendant du présent cours.

5 Supports de cours

5.1 Forums

Il y a plusieurs forums de discussion sur le site du MOOC (Coursera) et sur le site Moodle (EPFL). Celui sur Moodle (EPFL) est destinés aux personnes souhaitant poser des questions sur le contenu du cours (au sens large). Il a pour but principal de vous permettre d'interagir entre vous et avec l'équipe du cours et poser vos questions sans avoir à attendre les séances « officielles ».

Si par contre vous avez des questions relatives au MOOC, à l'apprentissage du C++ en général, aux devoirs notés sur le MOOC (appelés (« *Exercices de programmation* »/« *Programming assignment* » sur la plate-forme Coursera), veuillez alors utiliser les forums du MOOC (Coursera).

Dans tous les cas, **n'hésitez pas à utiliser ces forums** ; ce sont des outils qui peuvent être très riches. Dites-vous bien que si vous vous posez une question, il y a sûrement au moins dix autres de vos camarades (de classe, et 1000 sur le MOOC) qui se posent la même question !

De plus, **n'hésitez pas à répondre aux questions de vos camarades**. Il n'y a pas de meilleur moyen d'apprendre que d'expliquer les concepts à d'autres. Sur Moodle, les membres de l'équipe peuvent « Approuver » (« Endorse » en anglais) les réponses d'autres étudiants, ce qui permet de savoir qu'elles sont considérées comme bonnes réponses.

Ainsi, si vous rencontrez des difficultés à résoudre un exercice pendant la semaine, je vous encourage vivement à décrire votre problème, si basique soit-il, pour que nous vous aidions à le surmonter. Il n'y a pas de question ridicule, même si certaines autres personnes semblent nettement meilleures que vous (ce ne sont d'ailleurs pas toujours celles qui finissent le mieux l'année !). Et tout le monde pourra profiter de la réponse !

Dans tous les cas, **n'**utilisez **pas** les emails personnels pour nous contacter (assistantes, assistants et enseignant), mais utilisez les forums pour cela. Donc, sauf urgence vraiment personnelle, n'envoyez **aucun message personnel**.

Quelques remarques importantes au sujet des forums :

- Des consignes d'utilisation du forum sont postées sur ce dernier en début de semestre. Lisez-les attentivement.
- Lisez régulièrement les forums car toutes les informations importantes relatives au cours y seront postées : dates, salles et consignes pour les tests, informations sur les cours, notes, changement éventuel au niveau de l'organisation, etc.
- Le forum ne fonctionne malheureusement habituellement qu'assez tardivement à plein régime. Souvent, par manque de confiance, ce moyen n'est que trop peu utilisé au début du semestre d'automne. C'est dommage car il peut vous épargner bien des heures de blocage face à une erreur de programmation.

  Je le répète donc, une seule consigne : n'hésitez pas à **y poser vos questions** et à **répondre à celles des autres**, et ce dès les premiers cours.

5.2 Ordinateurs

Via 150 « thin clients » (postes de travail) situés dans les salles CO020, CO021 et CO023, ou des connexions à distance depuis votre propre ordinateur (salles CO5, INM 10 et INM 11), vous aurez accès chacun et chacune à une machine virtuelle Linux (Ubuntu) tournant sur des serveurs (gros ordinateurs dédiés à cela). Il s'agit des ordinateurs officiels de ce cours (et qui servent aussi à d'autres enseignements).

Les assistantes et assistants seront à votre disposition dans ces salles pendant les séances d'exercices. Vous pouvez de plus accéder à ces salles quand vous voulez, à condition de ne pas déranger les cours qui s'y déroulent.

Vous pouvez aussi travailler sur votre propre ordinateur portable, via une « machine virtuelle » qui sera mise à disposition ou par accès réseau (plus d'explications dans la 1re série d'exercices).

Concernant l'accès aux machines virtuelles, les détails des comptes (ordinateurs et email) vous ont normalement été envoyés suite à votre inscription. Les accès aux salles sont attribués d'office aux étudiant-e-s de SMA. Les auditeurs et auditrices libres devront en faire la demande individuellement (venir me voir). En cas de problèmes d'accès aux salles ou aux ordinateurs, il faut contacter soit le Help-Desk (1234@epfl.ch, tél interne : 1234) si c'est un problème technique, soit le Service Académique (SAC) si c'est un problème administratif, typiquement un problème d'inscription.

Il est bien sûr impératif de respecter les directives relatives à l'utilisation des moyens informatiques de l'EPFL.

5.3 Transparents, séries et divers supports

La documentation du cours comprend les transparents, les séries d'exercices, quelques livres de référence, ainsi que des fiches résumé. Nous utiliserons également la documentation en ligne.

Les documents et les fichiers sont disponibles en ligne, sur Moodle. Vous pourrez ainsi *si nécessaire* en imprimer une version sur les serveurs d'impression de l'Ecole. Jusqu'à la date du cours, les transparents sont sujets à de petites modifications.

Il n'y a pas de polycopié pour ce cours, mais vous avez accès à tous les supports qui viennent d'être cités au travers du site du cours. Concernant la programmation, l'ensemble des exercices et corrigés sont accessibles sur le site du MOOC. Je vous conseille néanmoins d'acheter le livre d'exercices (de programmation), non pas parce que cela donne quelques menus droits d'auteurs à mes collègues, mais surtout parce que cela vous offre un matériel retravaillé, mieux structuré et mieux rédigé, le tout sur un support relié et compact. Mais libre à vous de voir.

À noter également que les PPUR ont publié des « BOOCs », totalement gratuits, qui sont des résumés format eBook des vidéos du MOOC :

`https://www.epflpress.org/produit/805/9782889143962/initiation-a-la-programmation-en-c`

5.4 Fiches résumé

Le but des fiches résumé est double : présenter de façon condensée ce qu'il faut connaître, puis avoir un accès très rapide à tel ou tel détail de syntaxe, une fois les concepts connus.

Note : les fiches résumé sont également incluses dans le livre d'exercices, au début de chaque chapitre.

5.5 Livres de référence

Ouvrage de référence pour la partie théorie du cours :

— Schiper, André. Découvrir le Numérique : Une Introduction à l'informatique et Aux Systèmes de Communication. 1re éd. Lausanne : Presses Polytechniques et Universitaires Romandes, 2016.

Print.
— Disponible à l'achat à La Boutique de l'EPFL
— Disponible en consultation en ligne via la bibliothèque de l'EPFL :
`https://go.epfl.ch/decouvrir-le-numerique`

Recueil d'exercices et fiches résumé pour la partie programmation :

— Chappelier, Jean-Cédric, et Seydoux, Florian. C++ Par la Pratique : Recueil d'exercices Corrigés et Aide-Mémoire. 4e éd. Lausanne : Presses Polytechniques et Universitaires Romandes, 2017. Print.
— Disponible à l'achat à La Boutique de l'EPFL
— Disponible en consultation en ligne via la bibliothèque de l'EPFL :
`https://go.epfl.ch/cpp-par-la-pratique`

6 Organisation du travail

6.1 Considérations générales et conseils

J'ai bien conscience que « cette branche n'est qu'une branche pratique », que « vous n'êtes pas en Section Informatique », etc. Et il est bien clair pour moi que le travail doit rester proportionnel à l'importance de la matière pour la filière concernée (c.-à-d. plus concrètement à son coefficient au plan d'études). Mais apprendre la programmation ne peut se faire sans un minimum d'investissement personnel, lequel peut représenter une charge assez lourde, sans être excessive. En organisant correctement leur travail sur l'*ensemble du semestre*, chaque année un nombre important d'étudiantes et étudiants arrivent très bien à gérer cette charge de travail et réussissent cette branche, ainsi que les autres plus importantes pour leur Section.

Le fait que certaines étudiantes et étudiants aient la perception de trop travailler dans cette branche, voire que d'autres passent effectivement trop de temps dessus, et pire !, au détriment des autres branches, provient de trois causes principales :

1. une mauvaise gestion des contraintes et des priorités ;
2. une mauvaise répartition du travail dans le temps (et, au second semestre, au sein du binôme du projet) ;
3. une mauvaise évaluation du travail à fournir.

Il me paraît donc extrêmement important que vous vous fixiez des objectifs (raisonnables) et organisiez correctement votre travail, *dès le début* et tout au long du semestre. N'hésitez pas à me demander conseil dans ce sens si nécessaire ; mais pour résumer :

1. Pour bien apprendre la programmation, il faut *pratiquer*. Cela ne se fait pas du jour au lendemain et demande en effet une certaine quantité de travail *et* une certaine régularité.
2. Néanmoins cette quantité de travail doit rester « raisonnable » (entre 9 et 14 heures *grand maximum* par semaine *TOUT COMPRIS* (cours (dont MOOC) + exercices + travail personnel)). [4]
3. Il s'agit là d'une branche de semestre pour laquelle vous ne travaillez plus du tout après la fin du semestre. La période sur laquelle vous devez fournir le travail pour cette branche étant plus courte, il est évident qu'à charge totale égale, la charge par semaine devient plus grande.
4. Les « Exercices de programmation » sur Coursera sont corrigés automatiquement. Cela a l'avantage de vous donner un feedback individuel. Cela a aussi le désavantage de présenter un cap *tout ou rien. Ne passez pas un temps incontrôlé* à vouloir à tout prix satisfaire le correcteur automatique ! Sachez vous arrêter lorsque le temps passé sur un problème excède le bénéfice que vous en retirez. Il est obligatoire de *soumettre* chaque exercice de programmation, mais *pas* d'en obtenir tous les

4. Je rappelle que l'EPFL considère que pour un cours 3+3 comme celui-ci, vous devez fournir une charge de travail personnel à la maison de 6h (en *plus* de ces 3+3h, donc !) : 1h au plan d'étude $\simeq$ 1 ECTS $\simeq$ 30h de travail sur le semestre.

points ! En particulier, l'exercice des lapins et des renards (vous saurez de quoi je parle) a de quoi se casser les dents si on ne sait pas s'arrêter.

5. Fixez-vous un objectif atteignable pour votre niveau : quel est votre objectif ? 6.0 à tout prix ? À quoi bon, si c'est pour ensuite rater l'Analyse...

   (Ceci dit, pour obtenir finalement un 4.5, il vaut mieux viser au-dessus (p.e. 5.0). Faire et rendre un exercice ne signifie pas nécessairement le réussir à 100% et obtenir tous les points.)

6.2 Proposition d'un plan général de travail

Comme j'ai bien conscience que c'est certainement une forme d'enseignement que vous n'avez pas encore pratiquée, je vous propose l'organisation suivante afin de bénéficier au mieux de l'outil pédagogique qu'est la « classe inversée » que constitue la partie programmation :

1. quelques jours *avant* le cours en auditoire (jeudi 9h15) : regarder la vidéo du MOOC et faire les quiz (dans et hors vidéo) ;

   temps de travail estimé : entre 1h30 et 2h30 ;
2. après avoir vu les vidéos et *avant* la séance d'exercices en classe (jeudi 10h15–12h00) : finir les quiz (si ce n'est pas fait) et commencer des exercices (fichier PDF) ; je vous conseille *vraiment* de commencer les exercices avant de venir en séance ;

   temps de travail estimé : entre 30 min. et 1h30 ;
3. jeudi 9h15–10h00 : participer au cours *ex cathedra* portant sur la matière de la semaine *écoulée*, poser des questions, répondre aux questions et participer aux discussions ;

   temps de travail : 45 min. ;
4. jeudi 10h15–12h00 : séance d'exercices en présence des assistantes ;

   temps de travail : 90 min. ;
5. entre le jeudi 12h00 et le moment où vous passez à la vidéo de la semaine suivante : faire et rendre les exercices corrigés du MOOC (« *Exercices de programmation* »/« *Programming assignment* » dans la terminologie Coursera) ;

   temps de travail estimé : entre 30 min. et 1h30.
6. pour le cours de théorie (vendredis) : si vous le souhaitez, quelques jours *avant* le cours en auditoire (vendredi 13h15) : lire les sections du livre concernant la semaine en cours ; cela est optionnel mais peut vous donner une longueur d'avance sur le contenu du cours ;

   temps de travail estimé : entre 45 min. et 1h30 ;
7. vendredi 13h15–15h00 : participer au cours *ex cathedra*, poser des questions, répondre aux questions et participer aux discussions ;

   temps de travail : 90 min. ;
8. vendredi 15h15–16h00 : séance d'exercices en présence des assistants ;

   temps de travail : 45 min. ;
9. avant le jeudi suivant : faire encore quelques exercices

   temps de travail estimé : entre 30 min. et 2h00.

Je vous propose un plan détaillé semaine par semaine en annexe A.

Remarque importante à propos des vidéos : vous avez certainement l'habitude de regarder des vidéos en ligne. Pour des vidéos **de cours**, je vous conseille *très fortement* de prendre des notes en même temps, comme si vous suiviez un cours normal. Ce processus de prise de notes vous aide à mémoriser (et plus tard à réviser) et vous force à synthétiser. Vous n'êtes plus une étudiante ou étudiant passif devant la vidéo, mais acteur de votre apprentissage.

7 Le mot de la fin... ou plutôt du début !

Je le répète donc, une des clefs essentielles de la réussite à ce cours est la **régularité** de votre travail et de votre progression. Mes collègues ont constaté avec regret lors des années précédentes que certaines et certains d'entre vous ne commencent réellement à travailler cette matière qu'à l'approche du premier examen. Il est souvent alors déjà trop tard pour combler l'importance des lacunes et les résultats s'en ressentent gravement. N'attendez donc pas la semaine d'interruption des cours pour nous faire part de vos éventuelles difficultés. Nous sommes à votre écoute *dès le départ* pour vous aider à les surmonter.

Il ne me reste maintenant plus qu'à vous souhaiter un bon apprentissage.

A Annexe : plan de travail détaillé

Avant de vous proposer un plan de travail détaillé semaine par semaine, voici la vue globale de ce semestre pour le MOOC et les deux parties du cours :

		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)		

Le MOOC a été conçu sur 8 semaines pour un travail de 5 à 7 heures par semaine, ce qui en terme de travail correspondrait à la partie pratique de ce cours ICC, mais sur 7 semaines uniquement. Les deux sujets délicats (fonctions et structures de données) sont étalés sur plusieurs semaines afin de vous offrir plus de pratique et présenter plus de compléments en cours. Cela engendre un léger décalage entre ce cours-ci et le calendrier du MOOC à partir de la 6e semaine du cours (semaine 5 du MOOC).

Pour faciliter l'organisation de votre apprentissage, je vous propose le plan détaillé, thème par thème, suivant.

La première semaine commence sur les chapeaux de roues. Il est **nécessaire** de regarder les vidéos de la Semaine 1 du MOOC **avant le premier cours**, sans quoi il vous sera très compliqué de réaliser la première série d'exercices encadrés.

A.1 Semaine 1 (8–12 sept.)

- avant jeudi : prendre connaissance du site du MOOC
  `https://www.coursera.org/learn/init-prog-cpp`
  s'y inscrire, et regarder les vidéos de la Semaine 1 du MOOC (*y compris* la section « Bases de programmation » qui n'est pas déployée par défaut) ;
  si vous souhaitez travailler directement sur vos ordinateurs portables, tenter d'installer les outils en suivant les informations du « Préambule » sur le MOOC
- jeudi 9–10 : présentation générale du cours en auditoire CO 3
- jeudi 10–12 : faire toute la série 1 de Moodle (« Prise en main ») :
  `https://moodle.epfl.ch/mod/page/view.php?id=1300158`
  puis la série d'exercices de la semaine 1 du MOOC
- vendredi 13–15 : introduction de la partie théorie du cours, et introduction sur l'algorithmique, en auditoire BCH 2201
- vendredi 15–16 : exercices sur la partie théorie
- vendredi–mercredi : faire des exercices supplémentaires « à la maison », et regarder les vidéos de la Semaine 2 du MOOC

A.2 Semaine 2 (15 sept.–…)

- avant jeudi matin : avoir vu les vidéos de la semaine 2 du MOOC et fait les quiz

— avant jeudi matin : avoir au moins essayé de faire quelques exercices de la semaine 2 du MOOC (fichier PDF)
— jeudi 9–10 : cours de restructuration en CO3 sur la première semaine du MOOC
— jeudi 10–12 : continuer les exercices du MOOC (fichier PDF)
— après la séance d'exercices : faire le premier devoir à rendre (« *Exercice de programmation* ») sur le MOOC
— vendredi 13–15 : cours sur la partie théorie du cours en auditoire BCH 2201
— vendredi 15–16 : exercices sur la partie théorie
— vendredi–mercredi : faire des exercices supplémentaires « à la maison », et regarder les vidéos de la Semaine 3 du MOOC

A.3 Semaine 3

Essentiellement le même schéma que la semaine 2. Nous sommes maintenant en régime de croisière.

A.4 Semaine 4

Même schéma que la semaine 3.

A.5 Semaine 5

Le schéma reste, dans les grandes lignes, le même que les semaines précédentes, sauf que le sujet traité sur une semaine dans le MOOC de programmation (« *fonctions* ») est cette fois étalé sur deux semaines de cours, afin de vous laisser le temps de bien comprendre et bien travailler (faites deux fois plus d'exercices) le sujet correspondant.

A.6 Semaine 6

La démarche de travail reste similaire à la semaine 5 (sur le sujet des « *vectors* »), mais en raison de l'approche de l'examen (semaine 7), la séance d'exercices de théorie est dédiée aux révisions. Les exercices du cours de cette semaine seront donnés en semaine 8 (suite du sujet commencé cette semaine).

A.7 Semaine d'interruption des cours

L'occasion de rattraper du retard que vous auriez accumulé dans vos diverses matières. Mais aussi l'occasion de vous reposer et reprendre des forces.

A.8 Semaine 7

Schéma usuel pour la partie programmation. Par contre, **attention !**, **vendredi 31 octobre (à confirmer) de 13h15 à 16h00** : **premier examen** du cours.

Il porte sur les quatre cours du module I de la partie théorie et sur les semaines 1 à 6 (« vectors ») de programmation.

A.9 Semaines 8 et 9

Schéma usuel, comme les semaines 3 à 5, à la différence que les quatre sujets `vector`, `array`, `string` et `struct` traités en deux semaines sur le MOOC sont ici étalés sur trois semaines (y compris la semaine 7).

A.10 Semaines 10 et 11

Les cours de programmation de ces deux semaines ne font pas partie du MOOC, mais le schéma d'apprentissage reste le même.

Les vidéos et séries d'exercices sont disponibles sur Moodle.

A.11 Semaine 12

Les exercices de programmation sont dédiés à des révisions.

A.12 Semaine 13

Il n'y a pas de cours de programmation, mais le cours du jeudi 11 décembre porte sur la partie « Sécurité ».

La série d'exercices de programmation est dédiée à l'étude de cas du MOOC.

A.13 Semaine 14

Le **vendredi 19 décembre (à confirmer) de 13h15 à 16h00** a lieu l'**examen final** qui porte sur *l'entièreté* du cours, parties théorie et programmation.