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+SPH (CS-119(d))
J.-C. Chappelier

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

Présentation générale du cours

IX MMXX

1 Introduction

Ce document a pour but de vous informer sur la pédagogie du cours « Information, Calcul, Communication » donné aux Sections MA et PH, 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 clé secrète, RSA).

La partie pratique vise à :
- enseigner les notions fondamentales communes à la plupart des langages de programmation généralistes et « orientés objet » (variables, expressions, structures de contrôle, fonctions, entrées-sorties, ...)
- 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 « Programmation Orientée Objet » (CS-112(g)).

Ce cours est en premier lieu conçu et organisé **pour les débutants** 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 programmeurs *compétents*.

Pour les notions plus avancées, ce cours a aussi pour objectif de *consolider l'acquis* des non-débutants, en y apportant les bases plus formelles qui font parfois défaut dans un apprentissage autodidacte (méthodologie, notions algorithmiques, bonnes pratiques, ...).

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

Ce principe a conduit à introduire plusieurs éléments :
- un **accès diversifié** au contenu : transparents du cours, exercices, fiches résumé, mini-références, livre conseillé et références externes (« en ligne » et bibliographiques) sont les différents supports mis à disposition ;
- un **accès hiérarchisé par niveau** des élèves : deux niveaux (standard et avancé) pour le contenu du cours (transparents) et quatre niveaux pour les exercices (voir section 3.3.2).

Il est donc primordial que vous identifiez clairement les éléments qui vous sont destinés (débutant ou avancé) :
- pour les débutants : ne vous laissez pas noyer avec des notions trop avancées ;
- pour ceux qui pensent déjà connaître un sujet traité : ne vivez pas trop sur vos acquis et ne vous laissez pas dépasser le moment venu.

Pour cela, différentes indications de niveaux sont données : icône « avancé » dans les transparents, 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 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, …) 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.). Mais ce n'est ni un cours d'algorithmique, ni un cours de génie logiciel, seulement un avant-goût de ces disciplines fondamentales pour l'informaticien.

Pour remplir les objectifs multiples de ce cours, les outils pédagogiques suivants sont mis à votre disposition :

— le site du cours sous Moodle :
`http://moodle.epfl.ch/course/view.php?id=14023`
sur lequel vous trouverez un accès aux transparents du cours, aux séries d'exercices et à leur corrigé 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 (fiches résumé et mini références ; voir section 5.4) ;
— des forums de discussion : sur 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 :
— la partie théorie : deux heures de cours en auditoire le vendredi après-midi, suivi d'une heure d'exercices sur papier juste après ;
— la partie pratique, détaillée ci-dessous.

La partie pratique recourt à *deux* formes **complémentaires** d'enseignement, aussi importantes l'une que l'autre : un MOOC (sur Internet) et une partie présentielle (séances de cours ex cathedra en auditoire et séances d'exercices en salles informatiques).

Les paragraphes suivants expliquent leur rôle respectif et surtout *comment organiser votre travail* avec un MOOC (section 6).
Les **spécificités** de ce semestre **d'automne 2020** en raison des mesures de protection contre le Covid-19 sont ensuite expliquées section 3.4.

3.1 MOOC (Massive Open Online Course)

Un MOOC de *8 semaines*, « Initiation à la programmation (en C++) [1] », a été créé dans le but d'offrir de façon pédagogique, en particulier aux plus débutants, le minimum de bases nécessaires à tout apprentissage de la programmation.

Il constitue à mes yeux 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 ;
— des tutoriels, exercices reprenant des exemples du cours et dont le corrigé est donné progressivement au fur et à mesure de la donnée de l'exercice lui-même ;
ils sont conseillés comme un premier exercice sur un sujet que l'étudiant(e) ne pense pas encore assez maîtriser pour aborder par lui-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 clé fondamentale de votre apprentissage de la programmation ; ne la négligez pas ;
— des « devoirs », exercices à rendre et qui seront corrigés automatiquement.

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

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'Ecole. Je leur donne donc un caractère *obligatoire* : il est nécessaire que vous les ayez abordés avant la dernière semaine du semestre, mais je vous conseille vivement d'être à jour avant le premier examen (bon entraînement).
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 [2] un email m'indiquant avec quelle adresse vous vous êtes inscrit(e) sur Coursera.

Un dernier conseil pour un cours « inversé » comme celui-ci (= où il faut regarder le cours en vidéo avant de venir en classe) : agendez vous un/des créneau(x) hebdomadaire(s) fixe(s) pour regarder les vidéos et travailler la matière de votre coté.

3.2 Cours ex cathedra (c.-à-d. en amphi)

Les cours ex cathedra réunissent ~~tous les~~un tiers des [3] étudiant(e)s dans un auditoire, une heure par semaine pour la partie pratique/programmation et deux heures pour la partie théorie.

Durant les deux premiers tiers du semestre (pendant le MOOC), le but de des « cours ex cathedra » de la partie *pratique* (le jeudi, donc) est de compléter le MOOC en

1. reprenant si nécessaire les *principaux concepts fondamentaux* [4] ;
2. vous montrant, à l'aide d'exemples, comment résoudre des problèmes particuliers (« études de cas ») ;
3. discutant telle ou telle solution ;
4. répondant à vos questions [5].

Pendant le dernier tiers du semestre (après le MOOC), ces cours apporteront des compléments en présentant des thèmes non abordés dans le MOOC.

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) seront disponibles sur le site (EPFL) du cours en fin de semaine précédente. Des compléments oraux, sous forme d'exemples additionnels ou de discussions, sont souvent apportés au tableau pendant le cours ex cathedra.

Les examens portant sur la matière qui est enseignée dans les MOOCs **ET** dans les cours ex cathedra, il est important de bien connaître le contenu (et le style d'enseignement) pour arriver à bien se préparer, même si vous ne vous considérez plus comme un novice en programmation.

3.3 Pratique : séances d'exercices et travail à la maison

3.3.1 Organisation générale

Chaque cours est suivi d'une séance d'exercices (2020 : en présentiel ou à distance via Zoom) où les notions théoriques sont mises en pratique. Trois heures d'exercices sont prévues par semaine (une pour la partie théorique et deux pour la partie pratique), mais le temps requis pour résoudre les exercices peut varier, parfois considérablement, en fonction des connaissances préalables et de la préparation de chaque

2. `jean-cedric.chappelier@epfl.ch`
3. Voir section 3.4.
4. Il est donc absolument nécessaire que vous ayez vu les vidéos *AVANT* de venir en cours !
5. Il faut donc que vous en ayez !

étudiant(e). Il relève donc de la responsabilité de chacun d'entre vous de compléter ces séances par la quantité de travail « à la maison » appropriée.

Personnellement, je conçois qu'un(e) étudiant(e) moyen(ne) devrait consacrer trois à quatre heures *supplémentaires* de travail personnel par semaine, et un(e) étudiant(e) 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 calculette).

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 seul(e) ou avec des camarades, sur le matériel de l'école ou sur votre propre ordinateur portable.

Plusieurs assistants (et souvent 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 les assistants. Ils 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 sera mis à disposition sur le site (EPFL) du cours (et aussi sur le MOOC pour la partie programmation) en fin de semaine précédente. Le corrigé de la semaine précédente sera également mis à disposition à ce moment là. Tout le contenu de la partie programmation est également disponible dans mon livre « *C++ par la pratique* » (PPUR), en vente à la librairie « La Fontaine », dont une version électronique est également disponible.

Remarque importante : Il est vivement recommandé de participer aux séances d'exercices (2020 : que ce soit en présentiel ou par Zoom) :
- cela vous permet d'assurer la *régularité* de votre progression qui est une clé essentielle de la réussite au cours ;
- cela vous permet de bénéficier de l'aide des assistants ;
- 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 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 un débutant ; ceci afin que 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).

Concernant les exercices de programmation, je vous en fournis de deux types :
- des exercices généraux, dont une sélection a été mise à l'identique sur le site du MOOC ;
- 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, mais je les ai aussi regroupés sur cette page : `https://progmaph.epfl.ch/lien-icc.html`.

Je vous encourage 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 aspects du cours.

Dans le but de vous aider à vous organiser, les exercices sont organisés par niveau de difficulté (de 0 à 3) :
- **niveau 0** : reprise pas à pas d'un exemple du cours ; ils peuvent sans problème être sautés par

tous ceux qui estiment avoir une suffisamment bonne compréhension de la programmation de base et du cours du jour ; ils doivent par contre, à mon avis, être repris par les novices ;
ces exercices correspondent aux « Tutoriels » du MOOC ;

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

— **niveau 2** : ces exercices plus avancés devraient être abordés par 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 ;

— **niveau 3** : ces exercices d'un niveau *avancé* sont pour les plus motivés/habiles d'entre vous ; ils n'existent pas systématiquement dans chaque série ; 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 (sauf une par semestre, la « série notée », à la 10^{e} semaine) et il ne vous est pas demandé de les rendre. 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 impératif 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.

3.4 Spécificités automne 2020

Comme vous le savez, en raison des mesures de protection contre le Covid-19, un tiers seulement de la classe est autorisé à venir sur le campus (suivant le plan rappelé en annexe A). J'ai donc décidé d'adapter la façon d'enseigner ce cours de la manière suivante :

— concernant la partie pratique (programmation C++, des jeudis) :
  — pour le cours : ce cours étant déjà depuis plusieurs années sur MOOC, il n'y a pas trop de changement ;
  — les « études de cas » (1 h de cours les jeudis de 11 à midi) seront filmées, puis disponibles dans l'après-midi ;
  ce décalage ne devrait pas être gênant puisque le travail de cette matière s'étale de toute façon sur plusieurs jours (travail à la maison avec le MOOC) ;
  de plus, l'apport de ces séances dépend de chacun : j'ai chaque année des retours très contrastés : certain(e)s trouvant ces compléments et études de cas très utiles, d'autres estimant que ce n'est pas nécessaire ;
  — les séances d'exercices seront en présentiel pour un tiers de la classe (périodicité de 3 semaines

donc) et en ligne (meeting Zoom) pour les deux autres tiers, aux mêmes horaires ;
- concernant la partie théorique (des vendredis) :
  - pour le cours :
    - des semaines 1 à 4 incluses : le cours sera enregistré et, pour des raisons purement techniques, disponible en fin de journée ;
      des vidéos de collègues seront également disponibles *en avance* si vous préférez ;
    - à partir de la semaine 5 : le cours de l'an passé ayant été enregistré, il vous sera demandé de le regarder en avance à la maison et les 2 heures en classe seront mises à profit pour répondre à vos questions et proposer des « études de cas » ou correction dirigées d'exercices (sur tout le matériel passé, c.-à-d. que nous couvrirons ensemble tout ce qui est nécessaire depuis la dernière fois que l'on se sera vus) ;
  - pour les séances d'exercices : comme pour la partie pratique, elles seront en présentiel pour un tiers de la classe (périodicité de 3 semaines) et en ligne (meeting Zoom) pour les deux autres tiers, aux mêmes horaires ;
- pour les deux parties, vous aurez de plus la possibilité de venir de façon volontaire et sans contrainte, tous les **samedis** (sauf 19/09, 31/10 et 19/12 ; cf annexe A), de 15h00 à 17h00, pour des séances pratiques avec assistants ; sachez en profiter !
  Les détails pratiques seront donnés sur le forum Moodle du cours.

Les détails, semaine par semaine, sont fournis en annexe A.

CONSEILS spécifiques à ce mode d'enseignement
- venez sur le campus quand vous y êtes autorisé(e)s ; sachez profiter de notre présence et de celles de vos camarades, pour échanger, poser des questions, etc.
- ne vous laissez pas distancer, ni par rapport à la matière (travaillez régulièrement toutes les semaines, même les jours où vous n'êtes pas sur le campus), ni socialement (participez aux séances en ligne),
- apprenez à travailler efficacement avec des vidéos : ne restez pas passif/passive, mais prenez des notes, faites des poses et questionnez vous/réfléchissez à ce qui vient d'être dit ; ce sera le meilleur moyen de vous approprier le contenu ;
- utilisez d'autres supports (papier) en complément, en particuliers les « BOOCs » pour la partie programmation et le livre du cours pour la partie théorie ;
- oser poser des questions, soit en présentiel (pendant la semaine ou aux séances optionnelles des samedis, soit dans les séances en ligne (meeting Zoom), soit par écrit dans le forum du cours ;
- agendez vous un/des créneau(x) hebdomadaire(s) fixe(s) pour regarder les vidéos et travailler la matière de votre coté.

4 Comment le cours est-il évalué ?

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

Cette année[6] Les connaissances que vous aurez acquises seront évaluées à l'aide de deux examens différents, tous sur papier uniquement :
- un examen de deux heures quarante-cinq le **vendredi 30 octobre 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 (fonctions) ; cet examen est « avec document » ;
- un examen final de deux heures quarante-cinq, le **vendredi 18 décembre de 13h15 à 16h00**, portant sur l'*entièreté* du cours, parties théorique et pratique ; cet examen est « avec document ».

La note finale du cours est calculée suivant la répartition :
- examen 1 : 40% ;
- examen final : 60%.

Plus précisément : soit p_x le nombre de points obtenus à l'épreuve x (0 en cas d'absence) sur un total

6. Attention pour les redoublants (ou pour les archives) : cela change par rapport aux années précédentes !

maximal pour cette épreuve de t_x ; la note publiée pour cette épreuve est alors l'arrondi suivant :

$$n_x = 1 - 0.25 \left\lfloor -20 \cdot \frac{p_x}{t_x} \right\rfloor$$

et la note finale N est ensuite calculée directement sur les points obtenus (et non pas les notes intermédiaires) par

$$N = 1 - 0.25 \left\lfloor -20 \cdot \frac{\sum_x \theta_x (p_x / t_x)}{\sum_x \theta_x} \right\rfloor$$

où θ_x est le coefficient de l'épreuve x (par exemple 0.6 pour l'examen final).

Enfin, le rendu de tous les devoirs du MOOC est obligatoire avant la dernière semaine du semestre, mais je vous conseille vivement d'être déjà à jour avant le premier examen (bon entraînement).
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). Ceux sur Moodle (EPFL) sont 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 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 !

Ainsi, si vous rencontrez des difficultés à résoudre un exercice pendant la semaine, je vous encourage vivement à nous 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 certain(e)s semblent nettement meilleur(e)s que vous (ce ne sont d'ailleurs pas toujours ceux/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 (assistants, 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, …
- Le forum ne fonctionne malheureusement habituellement qu'assez tardivement à plein régime. Souvent, par manque de confiance, ce moyen n'est que très 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 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 INF1 et INF 2), vous aurez accès chacun à une machine virtuelle Linux (Ubuntu 16.04) 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 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 bien 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 envoyées suite à votre inscription. Les accès aux salles sont attribués d'office aux étudiant(e)s de SMA et SPH. Les autres étudiant(e)s (auditeurs 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.

Vous comprendrez qu'il est impératif de respecter les directives relatives à l'utilisation des moyens informatiques de l'EPFL. Les jeux sont en particulier interdits dans toutes les salles (sauf ceux que je vous aurais demandé de programmer dans les exercices ☺).

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 et des mini-références. Nous utiliserons également la documentation en ligne.

Les documents et les fichiers seront disponibles en ligne quelques jours avant le cours en question. Vous pourrez ainsi *si nécessaire* en imprimer une version sur les serveurs d'impression de l'Ecole.

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 [7]. Ceci dit, bien que l'ensemble des exercices et corrigés soient accessibles sur le site du cours, je vous recommande néanmoins d'acheter le livre d'exercices, non pas parce que cela me donne quelques menus droits d'auteurs, mais surtout parce que cela vous offre un matériel retravaillé, mieux structuré et mieux rédigé, le tout sur un support relié et compact. Enfin, à vous de voir...

A noter également que les PPUR ont publié des « BOOCs », résumés format eBook des vidéos du MOOC :

`https://www.ppur.org/produit/805/9782889143962/Initiation%20a%20la%20programmation%20en%20C++`

5.4 Fiches résumé et mini-références

Le but des fiches résumé est double : présenter de façon condensée ce qu'il faut connaître, puis (plus tard pour les programmeurs) 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'exercice, au début de chaque chapitre.

Les mini-références visent par contre à donner une information plus complète sur les aspects techniques du langage C++ et offrent une présentation différente de la matière technique donnée dans le cours. Elles ne sont pas forcément faites pour un programmeur tout à fait débutant mais plutôt pour aller un peu

7. En raison des droits associés à leur publication dans le livre, l'accès Internet à ces exercices et corrigés est interdit hors domaine `.epfl.ch`. En pratique, cela signifie que pour y accéder depuis chez vous, vous devez établir une connexion authentifiée (via VPN).

plus loin. À vous de voir, suivant votre niveau, si elles vous conviennent. Mais il n'est pas anormal que certain(e)s les trouvent d'un niveau un peu trop avancé.

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 de l'année, chaque année un nombre important d'étudiant(e)s arrivent très bien à gérer cette charge de travail et réussissent cette branche ainsi que les autres plus importantes pour votre Section.

Le fait que certain(e)s étudiant(e)s 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 faire.

Il me parait donc extrêmement important que vous vous fixiez des objectifs (raisonnables) et organisiez correctement votre travail, *dès le début* et tout au long de l'année. 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)). [8]
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. Fixez-vous un objectif atteignable pour votre niveau : quel est votre objectif ? 6.0 à tout prix ? A quoi bon, si c'est pour ensuite rater l'Analyse...
   (Ceci dit, pour obtenir finalement un 4.5, il vaut mieux viser au-dessus (e.g. 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 de travail

Pour la partie théorie, il s'agit d'un cours usuel et je n'ai donc pas de conseil particulier à donner si ce n'est, peut être, de lire les transparents avant de venir en cours. Je pense que, même si vous ne les comprenez pas à ce moment là, cela vous prépare à mieux recevoir et comprendre le cours. Mais cela dépend de chacun.

Concernant la partie programmation, par contre, comme c'est une forme d'enseignement que vous n'avez encore certainement pas expérimentée, je vous propose l'organisation suivante afin de bénéficier au mieux de l'outil pédagogique qu'est le MOOC :

8. 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.

1. quelques jours *avant* le cours en amphi (jeudi 11h15) : regarder la vidéo 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 09h00–11h00) : finir les quiz (si ce n'est pas fait) et commencer des exercices ; je vous conseille *vraiment* de commencer les exercices avant de venir en séance ;
   temps de travail estimé : entre 30 min. et 1h30 ;
3. jeudi 9h15–11h00 : séance d'exercices ;
   temps de travail : 1h45 min. ;
4. jeudi 11h15–12h00 : assister au cours, poser des questions ;
   temps de travail : 45 min. ;
5. entre le jeudi 12h00 et le moment où vous passez à la vidéo de la semaine suivante : faire et rendre le devoir noté du MOOC (« *Exercices de programmation* »/« *Programming assignment* » dans la terminologie Coursera) ;
   temps de travail estimé : entre 30 min. et 1h30.

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

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

Je le répète donc, une des clés essentielles de la réussite à ce cours est la **régularité** de votre travail et de votre progression. J'ai constaté avec regret lors des années précédentes que certain(e)s d'entre vous ne commencent réellement à travailler cette matière qu'à l'approche de la série notée. Il est souvent alors déjà trop tard pour combler l'importance des lacunes et les résultats s'en ressentent gravement. N'attendez donc pas l'approche de la série notée 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	exercices prog. 1h45 Jeudi 9-11	cours prog. 45 min. Jeudi 11-12	cours théorie 90 min. Vendredi 13-15	exercices théorie 45 min. Vendredi 15-16	
1	17.09.20	--	0	prise en main	Bienvenue/Introduction	Introduction + Algo 1		18.09.20
2	24.09.20	1. variables	0	variables / expressions	variables / expressions	Algorithmes 1 (suite)		25.09.20
3	01.10.20	2. if	0	if – switch	if – switch	Algorithmes 2 (stratégies)		02.10.20
4	08.10.20	3. for/while	0	for / while	for / while	Calculabilité		09.10.20
5	15.10.20	4. fonctions	0	fonctions (1)	fonctions (1)	Représentations numériques		16.10.20
6	22.10.20		1	fonctions (2)	fonctions (2)	Echantillonnage + Filtrage		23.10.20
7	29.10.20	5. tableaux (vector)	1	vector	vector	**Examen 1 (2h45)**		30.10.20
8	05.11.20	6. string + struct	1	array / string	array / string	Echantillonnage + Reconstruction		06.11.20
9	12.11.20		2	structures	structures	Compression 1 (entropie)		13.11.20
10	19.11.20	7. pointeurs	2	pointeurs	pointeurs	Compression 2 (th. de Shannon)		20.11.20
11	26.11.20		-	pointeurs	entrées/sorties	Architecture des ordinateurs		27.11.20
12	03.12.20		-	entrées/sorties	entrées/sorties	Stockage/Réseaux		04.12.20
13	10.12.20		-	entrées/sorties	erreurs / exceptions	Sécurité 1		11.12.20
14	17.12.20	8. étude de cas	-	erreurs / exceptions	Théorie : sécurité 2	**Examen final (2h45)**		18.12.20
						(= classe « inversée »)		

Et vous trouvez sur la page suivante le détail des présences en fonction de votre groupe.

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. J'ai préféré étaler sur plusieurs semaines les trois sujets délicats (fonctions, tableaux et pointeurs) 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, lequel commence à la deuxième semaine du semestre).

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

A.1 Semaine 1 (14–20 sept.)

- avant jeudi : lire les messages Moodle et ce document
- jeudi 9–11 : faire *toute* la série 1 (Découverte du cours et de l'environnement Unix), en salle (**groupe A**) ou chez soi avec appui Zoom si nécessaire (**groupes B et C**)
- jeudi 11–12 : **groupe A :** présentation du cours en amphi CO1,
  groupes B & C : continuer la série 1, la vidéo du cours sera mise sur Moodle dès que possible (dans l'après-midi je pense, ce n'est pas moi qui contrôle le délai de publication)
- s'inscrire au MOOC (`https://www.coursera.org/learn/init-prog-cpp`)
- jeudi soir ou vendredi matin : regarder les transparents de la partie théorie
- vendredi 13–15 : **groupe B :** partie théorie du cours en amphi SG1
  groupes A & C : au choix : rien ou regarder la vidéo de collègues (proposée sur Moodle), la vidéo de mon cours sera mise sur Moodle dès que possible (en fin d'après-midi je pense)
- vendredi 15–16 : exercices sur la partie théorie (**groupe B :** en salles, **groupes A & C :** à la maison avec support Zoom)
- vendredi–mercredi : prendre connaissance du site du MOOC et regarder les premières vidéos (du MOOC),
  si nécessaire : regarder les vidéos des cours non encore vus (cours de jeudi 11–12 et vendredi 13–15)

Planning ICC		Groupe en présentiel sur le campus		
Semaine	dates	A	B	C
1	17/09		(à la maison)	(à la maison)
	18/09	(à la maison)		(à la maison)
	19/09	*pas de séance de support*		
2	24/09	(à la maison)	(à la maison)	
	25/09		(à la maison)	(à la maison)
	support 26/9	*optionnel*	*optionnel*	*optionnel*
3	01/10	(à la maison)		(à la maison)
	02/10	(à la maison)	(à la maison)	
	support 03/10	*optionnel*	*optionnel*	*optionnel*
4	08/10		(à la maison)	(à la maison)
	09/10	(à la maison)		(à la maison)
	support 10/10	*optionnel*	*optionnel*	*optionnel*
5	15/10	(à la maison)	(à la maison)	
	16/10		(à la maison)	(à la maison)
	support 17/10	*optionnel*	*optionnel*	*optionnel*
6	22/10	(à la maison)		(à la maison)
	23/10	(à la maison)	(à la maison)	
	support 24/10	*optionnel*	*optionnel*	*optionnel*
7	29/10		(à la maison)	(à la maison)
	Examen 30/10	**13h15 – 16h00**		
	31/10	*pas de séance de support*		
8	05/11	(à la maison)	(à la maison)	
	06/11		(à la maison)	(à la maison)
	support 7/11	*optionnel*	*optionnel*	*optionnel*
9	12/11	(à la maison)		(à la maison)
	13/11	(à la maison)	(à la maison)	
	support 14/11	*optionnel*	*optionnel*	*optionnel*
10	19/11		(à la maison)	(à la maison)
	20/11	(à la maison)		(à la maison)
	support 21/11	*optionnel*	*optionnel*	*optionnel*
11	26/11	(à la maison)	(à la maison)	
	27/11		(à la maison)	(à la maison)
	support 28/11	*optionnel*	*optionnel*	*optionnel*
12	03/12	(à la maison)		(à la maison)
	04/12	(à la maison)	(à la maison)	
	support 5/12	*optionnel*	*optionnel*	*optionnel*
13	10/12		(à la maison)	(à la maison)
	11/12	(à la maison)		(à la maison)
	support 12/12	*optionnel*	*optionnel*	*optionnel*
14	17/12	(à la maison)	(à la maison)	
	Examen 18/12	**13h15 – 16h00**		
	19/12	*pas de séance de support*		

A.2 Semaine 2 (21 sept.–…)

- avant jeudi matin : (point précédent) avoir vu les vidéos de la semaine 1 du MOOC et fait les quiz
- avant jeudi matin : avoir au moins essayé de faire quelques exercices de la semaine 1 du MOOC
- jeudi 9–11 : continuer les exercices (sur le site du MOOC ou dans la série d'exercices du site web EPFL (`http://progmaph.epfl.ch/`))
- jeudi 11h00 : **groupe C :** complément cours de prog. en CO1
  groupes A & B : continuer la série d'exercices, la vidéo du cours sera mise sur Moodle dès que possible
- jeudi–vendredi : continuer à faire des exercices de prog. (MOOC ou Moodle)
- après la séance d'exercices : faire le premier devoir à rendre (« *Exercice de programmation* ») sur le MOOC
- jeudi soir ou vendredi matin : regarder les transparents de la partie théorie
- vendredi 13–15 : **groupe A :** partie théorie du cours en amphi SG1
  groupes B & C : au choix : rien ou regarder la vidéo de collègues (proposée sur Moodle), la vidéo de mon cours sera mise sur Moodle dès que possible
- vendredi 15–16 : exercices sur la partie théorie (**groupe A :** en salles, **groupes B & C :** à la maison avec support Zoom)
- vendredi–mercredi : regarder toutes les vidéos de la semaine 2 du MOOC et faire les quiz
  si nécessaire : regarder les vidéos des cours non encore vus (cours de jeudi 11–12 et vendredi 13–15)

A.3 Semaines 3 et 4

Même schéma que la semaine 2 mais avec permutations des groupes.

A.4 Semaine 5

Le schéma reste, dans les grandes lignes, le même que les semaines précédentes, excepté deux changements notoire :

- 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
- les cours de la partie théorie (vendredi) ont été enregistré l'an passée et je vous demande donc de le visionner *avant* de venir en classe.
  À partir de cette semaine, les vendredis de 13 à 15, je ferais avec le groupe présent des « études de cas », réponses aux questions, etc. sur tous les sujets (de théories) traités depuis la dernière fois que j'ai vu le groupe (et donc, bien sûr, au départ à la 1ère séance avec un groupe donné : sur tous les sujets depuis le début de l'année).

A.5 Semaines 6 à 13

La démarche de travail reste similaire à la semaine 5 suivant le plan publié plus haut.

ATTENTION ! en semaine 7, **vendredi 30 octobre de 13h15 à 16h00** (notez bien le dépassement d'horaire !) a lieu le **premier examen** du cours. Il porte uniquement sur les quatre cours du module 1 de la partie théorie du cours et sur les semaines 2 à 6 (« fonctions ») de programmation.

A.6 Semaine 14

Il n'y a plus de cours de programmation, mais le cours du jeudi 17 décembre porte sur la fin de la partie « Sécurité ». Cette fin n'est pas sujet à examen.

Le **vendredi 18 décembre, de 13h15 à 16h00** a lieu l'**examen final** qui porte sur *l'entièreté* du cours, parties théorie et programmation.