

Cours 1.0

I. Présentation de l'unité

- I.1. Enseignant responsable et coordonnées : Denis BUREAU (5356, D.Bureau)
aussi responsable du bloc informatique du Tronc Commun
- I.2. Autres intervenants : [coordonnées](#)
Vous devez savoir qui est votre intervenant. Venir me voir sans délai en cas de problème.
- I.3. Séquencement (7 séquences IN101 + projet PR102)
[+ [Enseignement de la programmation dans le Tronc Commun, dont la II](#)]
[+ [Séquencement de l'unité](#)]
Ce premier cours de la séquence 1 = introduction ; le prochain rentre dans le vif du sujet ==> beaucoup plus dense.
- I.4. Évaluation : QCM+mini-écrit (1), QCM+écrit (2), écrit final (3)
- I.5. Objectifs
- bases de la programmation en langage de haut-niveau
 - approche objet
 - solution en français à un problème (à découper en sous-problèmes)
 - langage Java et quelques bibliothèques
 - pas seulement fonctionner, aussi "bien" programmé
 - expérience pratique et opérationnelle
- I.6. Approche pédagogique
- hypothèse de départ : personne ne sait programmer en java
 - les objets d'abord, apprentissage incrémental (pas tout d'une notion)
ne peut fonctionner qu'avec votre participation active (par exemple, lire compléments de cours sur la page web) et nécessite d'apprendre le vocabulaire en français et en Java (+ rigueur)
 - cours, td, tp, travail personnel, post-assistance (5356/mail/PC), projet
Après un TP, **ne pas accepter de ne pas comprendre qqch** de dit ou d'écrit !
 - page web (retenir <http://www.esiee.fr/~bureau/unites.htm>) + polycopié + livre
 - polycopié = aide-mémoire, autre présentation (utile, mais pas la même progression pédagogique)
 - livre très pédagogique, non indispensable, couvre beaucoup plus de choses (3^{ème} ou 4^{ème} édition seulement, english is better).
- I.7. Votre rôle
- cours : ne pas faire de bruit, prendre des notes (je n'écris pas tout), poser des questions, relire (notes, résumé, poly), noter des questions pour les poser à la prochaine occasion
 - td : cours relu, apporter notes de cours, ne pas attendre de correction/essayer, poser des questions, terminer seul les exercices et les essayer sur machine (meilleure préparation pour les contrôles car écrire un programme sur papier, mais différence essentielle : td=mieux vaut écrire faux que rien alors que contrôle, surtout qcm=mieux vaut rien écrire que faux)
 - tp : cours relu, apporter notes de cours et TD, ne pas regarder une solution, poser des questions, terminer seul les exercices, demander de l'aide, partager le clavier (première moitié l'un écrivain, l'autre relecteur, puis seconde moitié le contraire ; les deux sont autant responsables de chaque erreur, l'un parce-qu'il l'a tapée, l'autre parce-qu'il ne l'a pas signalée)
- I.8. Environnement technique : PC, Linux, JDK1.6, BlueJ (presque tous les labos)
chez soi : Windows ou MacOS OK, tout gratuit

II. Introduction à la programmation

- II.1 Informatique (différents "métiers") : Utilisateur, utilisateur avancé, technico-commercial, technicien/dépanneur, programmeur*, concepteur*/analyste, chercheur/théoricien
Lire au moins les [19 premières pages](#) du premier chapitre de ce [long poly](#).
- II.2. Performances du matériel
- micro-processeur : 1GHz (à4) ==> environ 1ns
 - mémoire RAM : 1Go (à8), environ 10ns (à70) ==> 10x plus lent, (+ cache, - multi-processeurs, + multi-accès ...)
 - disque dur : 1To (à2), environ 10ms ==> 1 million x plus lent (==> charger, traiter, sauvegarder)
 - même les SSD : 128Go (à512) sont 10000 x plus lents que la RAM

[+ [Description d'un ordinateur](#)]

II.3. Langages de programmation

II.3.1. Il y en a beaucoup (400 au DOD en 1980) et on en invente presque toutes les semaines !

II.3.2. Niveaux : machine (binaire, 0/1), assemblage (ex: 68k, macro), haut-niveau ($X=Y+Z+1$)

II.3.3. Traducteurs (assembleur, compilateur, interpréteur)

II.3.4. "Philosophies" : impératif ou déclaratif (rare), fonctionnel, script, structuré ou non (rare), Orienté Objet ou non (classique)

II.3.5. Exemples : Java, C (17 à 15%). C++, Php, Python, C#, Visual Basic (8 à 5%). lisp/scheme, camL, prolog, shell, perl, ruby, javascript, tcl/tk (outils en électronique), objective-C (iPhone), pascal/delphi, ...

II.3.6. Historique (C, C++, Java)

II.3.6.a. C : 1972, Bell Labs AT&T, pour écrire Unix/Linux, ANSI-C 1983, ISO-C 1990 puis 1999, bas niveau, non OO, compilé.

II.3.6.b. C++ : 1983, "C O.O.", compatible ANSI-C, ISO-C++ 1998 puis 2003, mieux mais bas niveau existe toujours, compilé.

II.3.6.c. [Java](#) : 1995, Sun Microsystems, O.O., Java non OO = C++ non OO = C, pas de bas niveau

- inventé pour internet, multiples versions, processeur virtuel = JVM

- source compilé en bytecode, puis interprété

- avantages = indépendance matérielle, code très petit

- inconvénients = lenteur (realtime, 20x, <2x, temps réel, militaire), JVM très grosse, lente à démarrer (pas un problème pour les serveurs)

II.3.6.d. Les éditions : SE (Standard), EE (Enterprise), ME (Micro), Card (cartes à puce)

II.3.6.e. Les versions de SE : 1.0 .. 1.4.2 | 1.5 1.6 v7? (open source)

- grosses modifications du langage entre 1.4.2 et 1.5

- JRE = Java Runtime Environment (=JVM, utilisation seulement) \neq \subset [JDK](#) = Java Development Kit (aussi développement)

- nombre de classes = 200 (v1) à 3700 (v6)

II.3.7. Pourquoi Java ?

- langage adapté à l'apprentissage (plus propre, moins permissif, signalement d'erreur, graphique, bibliothèque énorme)

- non OO identique à C/C++ (mais C reste indispensable sur certains kits et pour certaines applications)

- demandé dans les offres d'emploi ($>$ C/C++, standard pour développements web=JEE)

- Java et C \implies se mettre à C++ (3^{ème} langage moins difficile que 2^{ème})

[+ lire [chapitre 2](#)] [+ lire [popularité des langages](#)]

II.4. Algorithme (on effleure, voir cours en I2/I3/I4info)

II.4.1. Définition

- méthode systématique pour résoudre un problème en un temps fini

- différence avec les maths ?

- analogie culinaire : ingrédients, étapes, solutions multiples

- plusieurs algorithmes pour résoudre le même problème \implies choix, coût : performance/mémoire /matériel/lisibilité

II.4.2. [[Compromis espace/temps](#)]

II.4.3. Exemple de la multiplication (par un nombre à plusieurs chiffres)

II.4.3.a. [Algorithme classique](#)

II.4.3.b. [Algorithme "à la Russe"](#)

II.4.3.c. [Avantages](#)

II.4.3.d. [Optimisation](#)

----- à lire en travail personnel :

II.4.3.e. [Spécification](#)

II.4.3.f. [Traductions](#)

(Lire [pour les curieux](#))

Lire le poly :

pages situées avant le chapitre 1 et annexes 6 & 7

