INF3135 – Construction et maintenance de logiciels

Plan de cours – Hiver 2008

COORDONNATEUR		SALAH, Aziz	salah.aziz@uqam.ca	(514) 987-3000 1485	PK-4835
GROUPES	20	CHIEZE, Emmanuel	chieze.emmanuel@uqam.ca	(514) 987-3000 3699	PK-4115
		Mardi, de 18h00 à 21h00 (cours) – Lundi, de 18h00 à 20h00 (ateliers)			
	30	MALENFANT, Bruno	malenfant.bruno@uqam.ca	(514) 987-3000 4005	PK-4635
		Mercredi et vendredi, de 8h30 à 10h00 (cours) – Mercredi, de 10h30 à 12h00 (ateliers)			

DESCRIPTION

Initier les étudiants à la programmation à l'aide d'un langage impératif et procédural. Familiariser les étudiants à la construction professionnelle de logiciels et à leur maintenance.

Notions de base de la programmation procédurale et impérative en langage C sous environnement Unix/Linux (définition et déclaration, portée et durée de vie, fichier d'interface, stuctures de contrôle, unités de programme et passage des paramètres, macros, compilation conditionnelle). Décomposition en modules et caractéristiques facilitant les modifications (cohésion et couplage, encapsulation et dissimulation de l'information, décomposition fonctionnelle). Style de programmation (conventions, documentation interne, gabarits). Déboggage de programmes (erreurs typiques, traces, outils, par ex., gdb). Assertions et conception par contrats. Tests (unitaires, intégration, d'acceptation, boîte noire vs. boîte blanche, mesures de couverture, outils d'exécution automatique des tests, par exemple, xUnit, scripts). Évaluation et amélioration des performances (profils d'exécution, améliorations asymptotiques vs. optimisations, outils). Techniques et outils de base pour la gestion de la configuration (par exemple, make, cvs). Introduction à la maintenance de logiciels (types de maintenance, techniques de base, par exemple, remodelage, automatisation des tests de régression). Ce cours comporte une séance obligatoire de laboratoire (2 heures).

Préalables: INF2120 Programmation II

OBJECTIFS

Ce cours vise à introduire les étudiants à la construction professionnelle de logiciels. Il vise aussi à familiariser les étudiants avec la programmation procédurale et impérative.

À la fin du cours, l'étudiant devrait être capable :

- de développer et modifier des composants logiciels écrits dans un langage impératif et procédural;
- de bien maitriser le langage C et le compilateur du C sous UNIX/Linux;
- d'utiliser les notions de module, de cohésion et couplage, de complexité structurale, de dissimulation de l'information, etc., pour évaluer la qualité d'un composant logiciel;
- d'expliquer et d'utiliser les principales techniques de modularisation : décomposition fonctionnelle, dissimulation de l'information, filtres et pipelines;
- d'utiliser des assertions (pré/post-conditions, invariants) pour documenter des composants logiciels et assurer leur bon fonctionnement;
- de déboguer un programme à l'aide de techniques, stratégies et outils appropriés;
- de vérifier le bon fonctionnement d'un composant logiciel à l'aide de tests fonctionnels et structurels, et d'évaluer à l'aide des notions et outils appropriés la qualité des tests (par ex., complexité cyclomatique, mesures de couvertures des tests);
- d'évaluer de façon empirique à l'aide d'outils appropriés (par ex., profils d'exécution) les performances d'un composant logiciel de façon à pouvoir, si nécessaire, en améliorer les performances;
- d'utiliser divers outils (outil de gestion de configuration, fichiers makefile, langage de scripts) pour organiser le développement de programmes comportant plusieurs composants ou modules;
- d'expliquer les notions de base de la maintenance des logiciels et d'appliquer certaines techniques de maintenance (tests de régression exécutés automatiquement, remodelage de programmes).

ÉVALUATION

Description sommaire	Date	Pondération
Examen intra		30%
Examen final		30%
TP 1		10%
TP 2 et PTP 3 – 15% chacun		30%

Toute documentation est permise aux examens. Une moyenne d'au moins 50% aux examens est exigée pour réussir le cours.

Les travaux pratiques peuvent être réalisés individuellement ou en équipe de deux (2) personnes. Par contre, les examens doivent évidemment être faits de façon individuelle. Une pénalité de 10% par jour de retard sera

appliquée pour la remise des travaux pratiques. La qualité du français sera prise en considération (jusqu'à 10 % de pénalité).

Politique d'absence aux examens

Un étudiant absent à un examen se verra normalement attribuer la note zéro pour cet examen. Cependant, si l'étudiant était dans l'impossibilité de se présenter à l'examen pour un motif valable, certains arrangements pourront être pris avec son enseignant. Pour ce faire, l'étudiant devra présenter à son enseignant l'un des formulaires prévus à cet effet accompagné des pièces justificatives appropriées (par ex., attestation d'un médecin que l'étudiant était dans l'impossibilité de se présenter à l'examen pour des raisons de santé, lettre de la Cour en cas de participation à un jury).

Une absence pour cause de conflit d'horaires d'examen n'est pas considérée comme un motif valable d'absence, à moins d'entente préalable avec la direction du programme et l'enseignant durant la période d'annulation des inscriptions avec remboursement : tel qu'indiqué dans le guide d'inscription des étudiants, il est de la responsabilité d'un étudiant de ne s'inscrire qu'à des cours qui ne sont pas en conflit d'horaire.

Pour plus de détails sur la politique d'absence aux examens du Département d'informatique et pour obtenir les formulaires appropriés, consultez le site web suivant :
http://www.info.uqam.ca/enseignement/politiques/absence-examen

CONTENU

Contenu à titre indicatif

1 Présentation du plan de cours. Langage C : présentation générale, noms des variables et fonctions, instruction for, variables globales et visibilité.

2 Langage C (suite) : types de base, conversions, expressions et priorités des opérateurs, expressions conditionnelles, boucles, fonction main. Style de programmation.

3 Langage C (suite) : variables externes et statiques, préprocesseur, pointeurs et tableaux. Débogage.

4 Conception. Langage C (suite) : tableaux multi-dimensionnels, allocation dynamique de mémoire, arguments du programme.

5 Conception (suite). Langage C (suite) : structures, typedef, union.

6 Langage C (suite) : listes chaînées, fichiers et entrées/sorties standards.

Révision pour l'examen Intra.

7 Examen intra.

8 Gestion de la configuration : cvs, make. Outils Unix (scripts).

9 Programmation par contrats. Programmation défensive

10 Tests.

11 Tests (suite). Maintenance.

12 Évaluation des performances.

Révision pour l'examen final.

13 Examen final.

RÉFÉRENCES

V O Kernighan, B.W. et Ritchie, D.M. – *Le langage C* – **deuxième édition, Masson, Paris, 1990.**

V C Blaquelaire, J.P. – *Méthodologie de la programmation en C (3e édition)* – **Masson, 1998. [QA76.73 C15B75].**

V C Kernighan, B.W. et R. Pike, R. – *The Practice of Programming* – **Addison-Wesley, 1999.**

V C Kernighan, B.W. et R. Pike, R. – *La programmation - En pratique* – **Vuibert, 2001. [QA76.6K48814].**

V C Loukides, M. et Oram, A. – *Programming with GNU Software* – **O'Reilly, 1997.**

V C McConnell, S. – *Code Complete - A Practical Handbook of Software Construction* – **(Second edition). Microsoft Press, Redmond, WA, 2004.**

V C Purdy, G.N. – *CVS Précis & concis* – **Éditions O'Reilly, 2004.**

V C ZELLER, A. and KRINKE, J. – *Essential Open Source Toolset* – **John Wiley & Sons, Ltd, 2005.**

U R **http://www.info2.uqam.ca/~tremblay/INF3135**
Site web contenant des notes de cours et transparents, des anciens examens, etc.

A : article – C : comptes rendus – L : logiciel – N : notes – R : revue – S : standard – U : uri – V : volume

C : complémentaire – O : obligatoire – R : recommandé