



Département d'informatique
IFT 159 – Analyse et programmation

Plan de cours
Automne 2009

Enseignant**Gabriel Girard**

Courriel :	gabriel.girard@usherbrooke.ca
Local :	D4-1010-1
Téléphone :	(819) 821-8000 poste 63033
Site :	http://info.usherbrooke.ca/GabrielGirard/cours
Disponibilité :	sur rendez-vous

Horaire

Exposé magistral :	Mardi	13h30 à 15h20	salle D3-2035
	Jeudi	8h30 à 10h20	salle D3-2035
Exercices/laboratoires :	Mercredi	16h30 à 17h20	salle D3-2041

Description officielle de l'activité pédagogique¹

Objectifs	Savoir analyser un problème; avoir un haut degré d'exigence quant la qualité des programmes; pouvoir développer systématiquement des programmes de bonne qualité, dans le cadre de la programmation procédurale séquentielle.
Contenu	Introduction aux ordinateurs. Analyse et conception de solutions informatiques : simplification, décomposition, modularisation et encapsulation. Critères de qualité : la conformité, la fiabilité et la modifiabilité. Concepts de base de la programmation structurée : séquence, itération, sélection. Modélisation du traitement et modularité : concept de fonctions et d'abstraction procédurale. Concept de base de l'abstraction de données. Introduction aux concepts orientés objet : encapsulation, constructeur, destructeur, surcharge, notation UML. Récursivité. Processus logiciel personnel (PSPO). Revue de code. Tests unitaires.
Crédits	3
Organisation	3 heures d'exposé magistral par semaine 1 heure d'exercices par semaine 5 heures de travail personnel par semaine
Particularités	Aucune

¹<http://www.usherbrooke.ca/programmes/cours/IFT/ift159.htm>

1 Présentation

Cette section présente les objectifs spécifiques et le contenu détaillé de l'activité pédagogique. Cette section, non modifiable sans l'approbation d'un comité de programme du Département d'informatique, constitue la version officielle.

1.1 Mise en contexte

Le cours IFT 159, Analyse et Programmation, est le premier cours d'informatique des baccalauréats en imagerie et média numérique, en informatique, en informatique de gestion, en mathématiques et en physique. Il ne présuppose pas de connaissance en programmation. Il ne requiert que la connaissance de l'utilisation usuelle d'un ordinateur. Comme son nom l'indique c'est un cours d'analyse de problèmes et de programmation. L'analyse consiste à l'ensemble des activités dédiées à l'étude détaillée d'un problème. La programmation comprend des activités de conception, de codage, de test et de maintenance de programmes pour ordinateurs. Dans le cadre du cours IFT 159, nous traitons des notions d'analyse, de conception, de codage et de tests. Nous ne parlons pas de la maintenance.

Tout analyste-programmeur qui se voit confier le développement d'un système doit d'abord l'analyser au complet avant de penser à programmer la solution retenue. Cette approche est privilégiée lors des travaux. En effet, pour chaque travail vous devez remettre l'analyse, la conception, le code et les tests reliés à votre solution.

Ce cours est aussi le premier d'une chaîne de cours d'analyse de problèmes et de programmation: en S2 IFT 339, en S3 IFT 232, IFT 439 et IFT 359, puis éventuellement les cours de génie logiciel (IGL). De plus, les connaissances acquises en programmation dans ce cours sont primordiales pour les cours de IFT 209, IFT 215 et IFT 287.

1.2 Objectifs spécifiques

L'objectif du cours est d'apprendre à résoudre un problème en utilisant l'informatique. Le langage utilisé dans le cours est C++. Il est à noter que le but du cours n'est pas d'apprendre un langage mais bien le processus menant à résoudre un problème en utilisant l'informatique.

De façon plus précise, à la fin de cette activité pédagogique, l'étudiante ou l'étudiant sera capable :

1. de comprendre le fonctionnement d'un ordinateur dans le contexte de l'utilisation d'un outil d'élaboration d'une solution programmée ;
2. de lire et comprendre un énoncé de problème peu complexe et procéder à son analyse;
3. d'analyser un problème pour y offrir une solution algorithmique;
4. d'illustrer un algorithme en utilisant la représentation appropriée ;
5. de mettre en oeuvre un algorithme à l'aide d'un langage de programmation ;
6. de planifier des tests pour un programme ;
7. de vérifier le bon fonctionnement d'un programme ;
8. de respecter des normes et standards de programmation ;
9. d'utiliser les mécanismes élémentaires d'encapsulation orientée objet;
10. de rédiger un programme en appliquant les principes de base de la dérivation par enrichissements successifs.
11. de documenter la solution au moyen d'un document séparé (analyse et conception) ou d'une documentation incluse (programmation)

1.3 Contenu détaillé

Thème	Contenu	Heures	Objectifs	Travaux
1	Brève introduction aux ordinateurs : modèle pratique du calcul, modèle théorique du calcul, étapes de la mise en oeuvre et de l'exécution d'un programme.	3	1	
2	Développement de programmes : cycle de vie et phases de développement du logiciel; éléments de base du C++, notions de programmes (variables, constantes, types, énoncés, fonctions, ...), notions de compilation.	4	2..8,10,11	A/P
3	Analyse et conception descendante : phases de développement, spécification, analyse, conception, programmation, tests; introduction aux fonctions.	5	2..8,10,11	A/P
4	Structures sélectives des langages : expressions logiques; énoncé « if », énoncés composés, énoncé « if » emboîté, énoncé « switch » ; notions d'analyse et de conception.	5	2..8,10,11	A/P
5	Structures itératives des langages : concept d'itération; boucle conditionnelle et de comptage; analyse et conception; énoncés « while », « for » et « do ... while » ; récursivité; boucles emboîtées.	5	2..8,10,11	A/P
6	Les fonctions et introduction à la récursivité : concept de modularité; fiabilité des fonctions (exactitude, robustesse, assertions); utilisations avancées des fonctions; expressions logiques; récursivité; paramètres de sortie; tests; analyse et conception.	5	2..8,10,11	A/P
7	Organisation des données et types : représentation interne des données; création de nouveaux types simples; types énumérés; tableaux : concept et utilité des tableaux, tableaux à une dimension, tableaux à plusieurs dimensions, passage de tableaux en paramètre; types structurés (enregistrements); tableaux d'enregistrements; ensembles; utilisation d'une bibliothèque.	7	2..8,10,11	A/P
8	Introduction à l'abstraction de données : encapsulation; concepts d'abstraction de données; introduction aux classes; introduction à UML.	8	9	A/P
9	Récursivité : caractéristiques d'un problème récursif; exemples de problèmes récursifs.	3	3..5	
10	Introduction à la complexité algorithmique : définition; concept d'ordre de complexité; exemples.	2	3	
11	Conclusion.	1		

1. Les heures associées à un thème particulier inclues les heures d'exercices pour un total de 48 heures (12 semaines de quatre heures).
2. Le cours doit comprendre au moins cinq travaux pratiques couvrant tous les sujets marqués dans le tableau. Les lettres « A » et « P » représentent respectivement l'analyse/conception et la programmation.
3. Pour chaque travail, des séries de tests devront être remis ainsi qu'un estimé de l'effort requis pour effectuer le travail (PSP0).

2 Organisation

Cette section propre à l'approche pédagogique de chaque enseignante ou enseignant présente la méthode pédagogique, le calendrier, le barème et la procédure d'évaluation ainsi que l'échéancier des travaux. Cette section doit être cohérente avec le contenu de la section précédente.

2.1 Méthode pédagogique

Une semaine comprend quatre heures de présence en classe: trois heures de cours magistral et une heure d'exercices. Une heure est aussi prévue à l'horaire pour de l'assistance en laboratoire. Tous les thèmes du cours, sauf le premier, seront abordés de la même manière: au moins une étude de cas sera étudiée; l'analyse du problème sera effectuée et une fois correcte, la solution sera programmée; enfin on reviendra sur les éléments nouveaux du langage vus dans la programmation de la solution.

Chaque semaine, il y aura environ trois heures d'exposés magistraux décrivant la théorie ainsi que des exemples développés au tableau. Il y aura aussi environ une heure d'exercices qui seront directement intégrés dans les cours magistraux. Au moins une étude de cas sera abordée : analyse de la problématique, implémentation de celle-ci. On reviendra alors sur les éléments nouveaux de langage vus dans la programmation de la solution. Il y a une heure d'assistance en laboratoire prévue à l'horaire. Elle servira principalement à faire du soutien et du dépannage technique. C'est aux étudiants et étudiantes que revient la tâche de s'organiser pour pratiquer ce qui sera vu en cours.

2.2 Calendrier

Semaine	Date	Thèmes	Laboratoire	Lecture
1	31/08/09	1	Ouverture des comptes	Chapitre 1 de [5]
2	07/09/09	2	Présentation de l'environnement	Chapitre 2 de [5]
3	14/09/09	3	Au besoin	Chapitre 3 de [5]
4	21/09/09	3-4	Au besoin	Chapitre 3-4 de [5]
5	28/09/09	4-5	Au besoin	Chapitre 4-5 de [5]
6	05/10/09	5	Au besoin	Chapitre 5 de [5]
7	12/10/09	5-7 (tableau)	Au besoin	Chapitre 5 et sections 9.1, 9.2, 9.4, 9.5 et 11.1 de [5]
8	19/10/09	Intra.		
9	02/11/09	6-7 (tableau)	Au besoin	Chapitre 6, section 9.3 de [5]
10	09/11/09	7	Au besoin	Chapitres 7, sections 9.3, 9.7 et 9.8 de [5]
11	16/11/09	8	Au besoin	Chapitre 10 de [5]
12	23/11/09	8	Au besoin	Chapitre 10, section 11.7 de [5]
13	30/11/09	9-10-11	Au besoin	Chapitre 12, section 9.6 de [5]
14	7/12/09	Finaux		
15	14/12/09	Finaux		

2.3 Évaluation

Devoirs (6)	30 %
Intra	30 %
Final	40 %

Conformément aux articles 36, 37 et 38 du règlement facultaire d'évaluation des apprentissages² l'enseignant peut retourner à l'étudiante ou à l'étudiant tout travail non conforme aux exigences quant à la qualité de la langue et aux normes de présentation.

Le plagiat consiste à utiliser des résultats obtenus par d'autres personnes afin de les faire passer pour sien et dans le dessein de tromper l'enseignant. Si une preuve de plagiat est attestée, elle sera traitée en conformité, entre autres, avec l'article 8.1.2 du *Règlement des études*³ de l'Université de Sherbrooke. L'étudiant ou l'étudiante peut s'exposer à de graves sanctions dont automatiquement un zéro (0) au devoir ou à l'examen en question.

Ceci n'indique pas que vous n'avez pas le droit de coopérer entre deux équipes tant que la rédaction finale des documents et la création du programme reste le fait de votre équipe. En cas de doute de plagia, l'enseignant peut demander à l'équipe d'expliquer les notions ou le fonctionnement du code qu'il considère comme étant plagié. En cas de doute, ne pas hésiter à demander conseil et assistance à l'enseignant afin d'éviter toute situation délicate par la suite.

2.4 Échéancier des travaux

TP #	Réception du problème à analyser	Thème	Remise de l'analyse	Réception de l'analyse programmer à	Thème	Remise de la programmation
1	08/09/09	Analyse simple	14/09/09	14/09/05	Séquence Données simples	18/09/09
2	18/09/09	Diagramme structurel	25/09/09	25/09/09	Fonctions	1/10/09
3	1/10/09	Flots de données	8/10/09	8/10/09	Sélection et itération	15/10/09
4	15/10/09	Idem	3/11/09		Tableaux 1 dim. Récursivité	10/11/09
5	10/11/05	Analyse moyenne	17/11/05		Données composées Récursivité	24/11/05
6	24/11/09	Analyse orientée objet	30/12/09		Encapsulation	7/12/09

Directives particulières :

2.4.1 Évaluation

L'évaluation porte sur trois volets : un intra (30%), un final (40%) et des travaux (30%).

Examens

Les examens porteront sur toute la matière vue en classe y compris la programmation. À moins d'avis contraire, la consultation du manuel de cours et des notes de cours est autorisé à l'examen. Aucune ordinateur, téléphone cellulaire, calculatrice ou autre appareils électroniques n'est autorisé lors des examens.

Devoirs

1. Les trois premiers devoirs ont pour but de faire comprendre les principes de base de l'analyse de problème, de l'écriture d'une solution et de la programmation à l'aide du langage vu en cours (le C++). Ces premiers travaux sont tous divisés en deux phases. Ils consistent à :

(a) analyser un problème et

(b) programmer la solution mise de l'avant d'un autre problème, mais similaire au premier.

Les devoirs suivants ont pour but de vérifier que les étudiantes et les étudiants savent analyser un problème plus complexe, concevoir une solution et finalement implémenter leurs solutions.

2. Tous les travaux sont effectués par équipe de un ou deux étudiant(e)s. Tout travail effectué par équipes de trois se voit imposer une pénalité de 25%. La pénalité est de 100% pour des équipes de plus que trois.

²http://www.usherbrooke.ca/accueil/documents/politiques/pol_2500-008/pol_evaluation/sciences.html

³<http://www.usherbrooke.ca/programmes/etude>

3. La qualité du français et de la présentation est considérée dans le résultat du travail.
4. Les sujets des travaux seront disponibles au local du cours ou sur la page WEB du cours (page web du DMI sous la rubrique « Le personnel/Professeurs d'informatique/Gabriel Girard ») suivant qu'il y a ou qu'il n'y a pas de cours au jour donné. La remise du travail s'effectue le jour et à l'heure exigés. Le non respect de la date de remise entraîne une pénalité de 25% de la note par jour de retard, à moins d'un cas exceptionnel. Il est à noter qu'un oubli ou un emploi du temps chargé n'est pas un cas exceptionnel. Il en est de même en ce qui concerne une panne d'imprimante ou de votre modem. Cela signifie qu'il faut toujours viser à terminer son travail de programmation au moins 24 heures avant la date de remise pour tenir compte des pannes possibles et de la surcharge quasi-inévitable. Ceci est un conseil qui vaut son pesant de points. Les modalités de remise de chaque travail vous seront fournies avec le sujet de chaque travail.
5. Le respect des normes départementales est impératif. Un document contenant les normes du département en matière de programmation est fourni sur le site du cours à l'adresse <http://www.dmi.usherb.ca/~fraikin/cours/Normes/normes-de-programmation.pdf>
Cette contrainte permet de vérifier que l'étudiant ou l'étudiante sait s'astreindre à une discipline de programmation. Elle permet de plus de mieux insister sur les concepts importants du cours.
6. Pour réaliser la programmation chacun a besoin d'un code d'accès (numéro de compte & mot de passe) à l'ordinateur. Celui-ci est disponible dès la première semaine de cours. Pour obtenir votre numéro de compte, vous pouvez suivre les instructions fournies en annexe. Ces mêmes instructions sont affichées au laboratoire 1017 et sur la page WEB du Département d'informatique sous la rubrique « Ressources/Documentation ».
7. La programmation en C++ est réalisée dans l'environnement Unix, qui est la base du système Solaris sur les Suns. Elle peut être réalisée sous tout autre environnement (Windows, Linux ou Mac OSX par exemple), mais devra se trouver et fonctionner sous Unix au moment de la remise du travail. Il est dans votre intérêt d'apprendre à utiliser un minimum les Suns pour pouvoir tester vos programmes sur cette plate-forme.
De l'aide technique est disponible sur les ordinateurs des laboratoires, que ceux-ci tournent sous Windows XP, Linux ou Sun OS. Sur le site, les étudiants et étudiantes pourront trouver des liens et de la documentation pour travailler à partir de chez eux. Cependant, aucune autre aide ne sera fournie par manque de temps et de ressources. Il est conseillé, pour un travail hors des laboratoires, d'utiliser l'environnement de développement `code::blocks` qui est disponible sur la plupart des plates-formes.

3 Matériel nécessaire pour le cours

Les acétates du cours sont disponibles sur le WEB. Le manuel sur lequel est basé le cours est celui de Friedmann [5].

Les normes de programmation du Département d'informatique sont décrites dans [4]. Vous devez absolument vous procurer ce document et le lire. De même, toutes les informations nécessaires pour vous connecter à distance sur nos serveurs sont présentée dans [3].

4 Références

- [1] Yves BOUDREAU et Wacef GUERFALI : *C++ et un peu + (2e édition)*. Presses Internationales Polytechnique, 2001.
- [2] H. M. DEITEL et P. J. DEITEL : *C++ : How to program*. Prentice-Hall, 2001.
- [3] Benoit FRAIKIN : Connexion par SSH et la soumission par turnin. <http://www.dmi.usherb.ca/~fraikin/cours/SSH-turnin>, 2005.
- [4] Marc FRAPPIER : Normes de rédaction et de programmation du département. <http://www.dmi.usherb.ca/~fraikin/cours/Normes/normes-de-programmation.pdf>, 2005.
- [5] Frank L. FRIEDMAN et Elliot B. KOFFMAN : *Problem Solving, Abstraction, and Design Using C++*. Addison-Wesley, 2004.
- [6] Cay HORSTMAN et Timothy BUDD : *Big C++*. John Wiley, 2005.