UNIVERSITÉ DE SHERBROOKE

Département d'informatique

IFT359 — Programmation fonctionnelle

Plan de cours
Automne 2016

Enseignant

André Mayers
Courriel : *andre.mayers@USherbrooke.ca*
Local : D4-1018-2
Téléphone : (819) 821-8000 poste 62041
Site : *http://www.usherbrooke.ca/informatique/personnel/professeurs/professeurs/andre-mayers/*
Disponibilité : vendredi 10:30 à 12:00 + en tout temps via le *site web* du cours.

Professeur responsable

André Mayers

Horaire

Lundi	08:30 à 10:20	Salle : D3−2038 ou D4-1017
Jeudi	08:30 à 10:20	Salle : D3−2038 ou D4-1017

Description officielle de l'activité pédagogique[1]

Objectif : Formaliser les notions d'abstraction procédurale et d'abstraction de données dans le cadre de la programmation fonctionnelle.

Contenu : Qualité, modularité, conception fonctionnelle. Processus récursifs et itératifs. Objets atomiques. Listes. Abstraction d'ordre supérieur. Curryfication. Fermeture. Appels terminaux. Modèle d'exécution d'un programme fonctionnel. Application de la programmation fonctionnelle (structure de donnée non mutable, programmation par flots, appariement de forme …). Insistance sur la qualité de la solution.

Crédits : 3

Organisation : 3 heures d'exposé magistral par semaine,
1 heure d'exercices par semaine,
5 heures de travail personnel par semaine.

Antérieure: IFT 159

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. *http://www.usherbrooke.ca/fiches-cours/ift359*

1.1 Mise en contexte

Le cours IFT 359 introduit les étudiants à la programmation fonctionnelle. Celle-ci a débuté avec le langage Lisp créé par John McCarthy en 1958, dans les mêmes années où les langages de haut niveau supportant la programmation impérative (procédurale/structurée) tels que Fortran (1957) et Algol (1958) en étaient à leurs débuts.

Tandis que la programmation impérative est basée sur un modèle où un état change en fonction des opérations effectuées (machine de Turing), ce qui est très proche de l'implémentation physique (registres, mémoire, jeu d'instructions de l'UCT, etc.), la programmation fonctionnelle est basée sur un modèle mathématique (lambda calcul) dans lequel une fonction, souvent récursive, prend une entrée et calcule une sortie.

La programmation fonctionnelle est restée dans l'ombre de la programmation orientée objet (qui reste une forme de programmation impérative), mais elle refait maintenant surface, car ses propriétés sont très utiles pour faire face à certains problèmes actuels (e.g. LINQ, MapReduce).

De nombreux autres langages fonctionnels ont été développés depuis, en voici un aperçu :

1. Le langage Erlang (1986), développé par Ericsson, se base sur la programmation fonctionnelle pour offrir un langage propre à la programmation concurrente, parallèle et temps réel.
2. Le langage Scheme et ses dialectes (Racket, MIT Scheme), souvent utilisés dans le milieu académique, est un langage très bien supporté par de nombreux outils et des compilateurs efficaces.
3. Le langage Haskell (1990 et 1998) a été développé pour consolider les acquis des dernières décennies et se démarque par son typage statique totalement inféré, ses effets de bord contrôlés et son évaluation paresseuse.
4. Plusieurs langages fonctionnels sont fréquemment utilisés : JavaScript (1995) supporte de nombreux concepts de programmation fonctionnelle ; Autolisp (1986) est fréquemment utilisé par les ingénieurs à l'intérieur de Autocad ; et, EmacsLisp (1976) permet d'adapter Emacs à un grand nombre d'applications ou langage informatique.
5. Common Lisp (1984), OCaml (1996), Scala (2003) et F# (2008) sont des langages qui combinent la programmation fonctionnelle et la programmation orientée objet.
6. Plusieurs langages (Clojure, Scala et F#) ont appliqué la programmation fonctionnelle à des architectures logicielles existantes (la JVM et le.NET framework).
7. De plus en plus, C++ (11^{e}) et Java (8^{e}) permettent de développer avec un style fonctionnel.

1.2 Objectifs spécifiques

1. Connaître les principes théoriques : lambda-calcul, évaluation en ordre normal, évaluation en ordre applicatif, transparence référentielle, fonction d'ordre supérieur, curryfication.
2. Savoir implémenter un algorithme dans un langage fonctionnel : définition des identificateurs et des fonction, opérateurs logiques, instructions de contrôle, portée lexicale, structure de données basiques (liste, paire, etc.)
3. Savoir utiliser les notions clés de la programmation fonctionnelle (récursivité, fonction d'ordre supérieur, fermeture, appel terminal…).
4. Pouvoir expliquer le déroulement de l'exécution d'un programme fonctionnel : évaluation par environnement d'un programme fonctionnel.

5. Appliquer des techniques avancées de programmation en s'appuyant sur les propriétés des langages fonctionnels (e.g. évaluation paresseuse, création de nouvelles formes syntaxiques, création d'un langage embarqué (e.g. langage à objet), pattern matching).

1.3 Contenu détaillé

Thème	Contenu	Heure[a]	Objectifs	Travaux
1	Principes théoriques - lambda-calcul - fonction d'ordre supérieur - évaluation en ordre normal - évaluation en ordre applicatif - transparence référentielle - curryfication	6	1	TP1
2	Introduction au langage fonctionnel choisi - définition des identificateurs - définition des fonctions - portée lexicale - conditions et opérateurs de logique - instructions de contrôle - typage - structure de données basiques (liste, paire, etc.)	5	2	TP2
3	Évaluation par environnement d'un programme	4	4	TP3 & 4
4	Fonctions récursives	4	3, 4	
5	Fonctions d'ordre supérieur	4	3, 4	
6	Appel terminaux	4	3, 4	
7	Fermeture, affectation ou monade	3	3, 4, 5	
5 activités d'approfondissement parmi les 8 suivantes				
8	Approfondissement de la programmation fonctionnelle I : création de nouvelles formes syntaxiques	4	3, 5	TP5 & 6
9	Approfondissement de la programmation fonctionnelle II : Structure de données non mutable.	4	3, 5	

10	Approfondissement de la programmation fonctionnelle III : - Méta-évaluateur circulaire en ordre applicatif - Évaluation par environnement	4	3, 5	TP 5 & 6 (suite)
11	Approfondissement de la programmation fonctionnelle IV : - Méta-évaluateur circulaire en ordre normal, programmation par flots, évaluation paresseuse.	4	3, 5	
12	Approfondissement de la programmation fonctionnelle V : - création de langage embarqué (e.g. création d'un langage objet).	4	3, 5	
13	Approfondissement de la programmation fonctionnelle VI : utilisation explicite de continuation	4	3, 5	
14	Approfondissement de la programmation fonctionnelle VII : - Études des types : construction et preuves de programme.	4	3, 5	
15	Approfondissement de la programmation fonctionnelle VIII: appariement de forme.	4	3, 5	

a. Le nombre d'heure ci-dessous incluent les exercices dirigés.

2 Organisation

2.1 Méthode pédagogique

En moyenne, il y aura chaque semaine deux heures d'exposés magistraux décrivant la théorie ainsi que des exemples développés au tableau ou sur ordinateur. À chaque semaine, il y aura aussi deux heures d'exercices supervisés dans les laboratoires du département. Cet accent sur les exercices est motivé d'une part par les recherche sur la pertinence des pédagogies actives ; et d'autre part, parce que les concepts sont relativement faciles exprimés mais leurs applications en programmation exige une façon de "penser" peu mis de l'avant par les autres paradigmes de programmation.

Il y aura six travaux pratiques (TP), un examen intra et un examen final récapitulatif. Les travaux pratiques peuvent être faits en équipe de 2 personnes (ceux qui travailleront seuls seront évalués avec le même barème). Pour les TP, aucun retard n'est admis sauf pour les mêmes motifs que les absences aux examens (voir le règlement des études). L'évaluation tiendra compte de l'exactitude des sorties, mais aussi de l'utilisation judicieuse des concepts et du respect des standards de programmation. Une erreur de soumission entraîne automatiquement une pénalité d'au moins 10 % et le TP doit être reçu électroniquement à l'intérieur les 24 heures suivant l'heure de remise de ce TP .

2.2 Calendrier du cours

Voir le *site web* du cours.

2.3 Évaluation

Travaux (4 x 4% + 2 x 5%) :26%

Examen intra : 30%

Examen final : 44%

Votre participation au site web du cours, en particulier via les forums peuvent vous procurer jusqu'à 5 % de point boni.

2.4 Qualité du français et de la présentation

Conformément à l'article 17 du règlement facultaire[1] d'évaluation des apprentissages 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.

2.5 Plagiat

Un document dont le texte et la structure se rapporte à des textes intégraux tirés d'un livre, d'une publication scientifique ou même d'un site Internet, doit être référencé adéquatement. Lors de la correction de tout travail individuel ou de groupe une attention spéciale sera portée au plagiat, défini dans le Règlement des études comme « le fait, dans une activité pédagogique évaluée, de faire passer indûment pour siens des passages ou des idées tirés de l'oeuvre d'autrui. ». Le cas échéant, le plagiat est un délit qui contrevient à l'article 8.1.2[2] du Règlement des études : « tout acte ou manœuvre visant à tromper quant au rendement scolaire ou quant à la réussite d'une exigence relative à une activité pédagogique. » À titre de sanction disciplinaire, les mesures suivantes peuvent être imposées : a) l'obligation de reprendre un travail, un examen ou une activité pédagogique et b) l'attribution de la note E ou de la note 0 pour un travail, un examen ou une activité évaluée. Tout travail suspecté de plagiat sera référé au Secrétaire de la Faculté des sciences.

2.6 Directives particulières

Les travaux devront être remis électroniquement sur la page web du cours.

Les résultats des évaluations, les notes de cours, les directives des travaux, les exercices, les exemples faits avec DrRacket en classe seront placés sur le site web du cours. Pour les travaux, les exercices et les examens, il n'y aura pas de solutionnaires publics, mais vous pourrez consulter vos évaluations ou me montrer vos solutions pour en discuter lors des heures de disponibilité ou sur rendez-vous.

1. *http://www.usherbrooke.ca/accueil/documents/politiques/pol 2500-008/pol evaluation/sciences.html*
2. *http://www.usherbrooke.ca/programmes/references/reglement/discipline/#c4058*

3 Documentation

3.1 Manuel obligatoire ou notes de cours

Il sera de votre responsabilité d'imprimer les notes de cours et de les compléter en classe.

Les notes de cours sont accessibles sur la page web du cours :
https://www.usherbrooke.ca/moodle2-cours/course/view.php?id=11491

3.2 Bibliographie

1. Abelson H., Sussman G. et Sussman J. Structure and Interpretation of Computer Programs, 2e édition, Mc Graw Hill. (1996) *http://mitpress.mit.edu/sicp/*
2. Chazarain J. Programmer avec Scheme : De la pratique à la théorie, Vuibert. (1996).
3. Dybvig R. K. The Scheme Programming Language, Fourth Edition, Prentice Hall. (2009) *http://www.scheme.com/tspl/*
4. Friedman D. P., Felleisen M. The Little Schemer, MIT Press. 4th edition (1995).
5. Friedman D. P., Felleisen M. The Seasoned Schemer, MIT Press. (1995).
6. Felleisen M., Findler R. B., Flatt M. et Krishnamurthi S. How to Design Programs, MIT Press. (2003) *http://www.htdp.org/*