Mise à Niveau et Ouverture - STL

Cours 0

Emmanuel Chailloux

Parcours Science et Technologie du Logiciel
Master mention Informatique
Sorbonne Université

année 2020-2021

Informations sur le cours d'Ouverture

Sites

https://www-master.ufr-info-p6.jussieu.fr/2020/ouv
https://www-apr.lip6.fr/~chaillou/Public/enseignement/2020-2021/ouv
https://www-apr.lip6.fr/~pepin/

Equipe pédagogique

- cours : **Emmanuel Chailloux** (LIP6), **Martin Pépin** (LIP6)
- TME : **Emmanuel Chailloux** (LIP6), **Martin Pépin** (LIP6)
- Conférences : 5 conférenciers extérieurs

Description du cours

Deux parties séparées :

- une mise à niveau :
  - en algorithmique
    - analyse d'algorithme dans le pire des cas : diviser pour régner, programmation dynamique, algorithmes probablistes
  - et programmation
    - autour des langages fonctionnels typés
    - apprentissage OCaml
    - noyau fonctionnel, noyau impératif, modules simples, polymorphisme, structures dynamiques (listes, arbres, ...)
- des conférences d'ouverture sur des thématiques issues d'autres domaines (sociologie, économie, droit, histoire, ...)
  - conservation du développement logiciel de l'humanité
  - droits et informatique
  - sociologie de l'éco-système de la Silicon Valley
  - histoire des sciences et de l'informatique
  - métiers de la recherche (en informatique)

Déroulé et évaluation du cours

Le mardi après-midi :

- Mise à niveau :
  - 5 cours (du 29.9 au 27.10) le mardi de 14h-16h
  - + 5 TD/TME de 2h, le mardi de 16h-18h, décalés d'une semaine par rapport au cours
  - projet à rendre + rapport + soutenance = 70% réalisé en OCaml
  - soutenances les 1er et 8 décembre 2020
- 5 conférences :
  - mardi de 14h-16h
  - du 17.11 au 15.12
    - remise d'une synthèse d'une des conférence = 20%
    - présence obligatoire = 10%

Un mot sur OCaml (1)

- développé à l'Inria depuis 30 ans (`ocaml.org`)
- Langage de programmation **multi-paradigmes** (fonctionnel, impératif, objet, modulaire), de haut niveau avec des constructions **riches** et une **expressivité** accrue (foncteurs, types algébriques, lambdas, objets, exceptions ...)
- **Sûreté** augmentée par le typage statique avec inférence

```
# let rec map(f,l) = match l with
    []   -> []
  | t::q -> let nt = f(t) in
            let nq = map(f,q) in
              nt :: nq ;;
val map : ('a -> 'b) * 'a list -> 'b list = <fun>

# let compose(f,g,x) = f(g(x));;
val compose : ('a -> 'b) * ('c -> 'a) * 'c -> 'b = <fun>

# let addNL o = o#toString() ^ "\n";;
val addNL : < toString : unit -> string; .. > -> string = <fun>
```

Un mot sur OCaml (2)

- Ayant une réelle influence : F#, Swift, ReasonML
- des applications phares sur la fiabilité logicielle, l'implantation de langages, mais aussi en système, web et finance
- une communauté active, avec gestionnaire de paquets (opam)
- des compilateurs natifs et une machine virtuelle **légère**

```
# type couleur = Pique | Coeur | Carreau | Trefle;;
# type carte = As of couleur
             | Roi of couleur
             | Dame of couleur
             | Valet of couleur
             | Autre of couleur * int ;;
# let valeur couleur_atout cr = match cr with
| As _ -> 11
| Roi _ -> 4
| Dame _ -> 3
| Valet c -> if c = couleur_atout then 20 else 2
| Autre (_,10) -> 10
| Autre (c,9)  -> if c = couleur_atout then 14 else 0
| _ -> 0 ;;
val valeur : couleur -> carte -> int = <fun>
```

Logiciels du cours

- Langages
  - OCaml 4.xx (Inria)
    - pré-installé à la PPTI
    - à installer à la maison à partir du site `http://ocaml.org`
    - dans un navigateur avec TryOCaml : `http://try.ocamlpro.com/`
  - F# 4.xx (Microsoft) `http://fsharp.org/`
  - Swift 4.xx (Apple) : `https://swift.org/`
  - Reason (Facebook) : `https://reasonml.github.io/`
- Environnements de développement
  - mode Tuareg pour Emacs : `http://tuareg.forge.ocamlcore.org/`
  - Emacs et Merlin : `https://github.com/ocaml/merlin`
  - plug-in Ocaide pour Eclipse : `http://www.algo-prog.info/ocaide/`
  - VisualStudioCode : `https://code.visualstudio.com`
  - Atom script : `https://atom.io/packages/script`
  - Xamarin : `https://xamarin.com/studio`
  - Netbean : `http://ocamlplugin.loki-a.com/index.php`
- Environnement pédagogique
  - learn-ocaml

Bibliographie sur la partie programmation (1)

- livres sur OCaml
  - Xavier Leroy et al. The OCaml system : documentation and user's manual Inria, 2018.
  - Emmanuel Chailloux, Pascal Manoury et Bruno Pagano. Développement d'Applications avec Objective Caml. O'Reilly, 2000.
  - Guy Cousineau et Michel Mauny. Approche fonctionnelle de la programmation. Dunod, 1995.
  - Philippe Nardel. Programmation fonctionnelle, générique et objet : Une introduction avec le langage OCaml. Vuibert, 2005
  - Pascal Manoury. Programmation de droite à gauche, et vice-versa. Paracamplus, 2012.
  - Sylvain Conchon et Jean-Christophe Filliâtre. Apprendre à programmer avec OCaml. Eyrolles, 2013.
  - Yaron Minsky, Anil Madhavapeddy, Jason Hickey - Real World OCaml - O'Reilly - 2013

Bibliographie sur la partie programmation (2)

- cartes de référence OcamlPro :
  - langage :

    `http://www.ocamlpro.com/wp-content/uploads/2019/09/ocaml-lang.pdf`
  - bibliothèque :

    `http://www.ocamlpro.com/wp-content/uploads/2019/09/ocaml-stdlib.pdf`
- cours :
  - programmation fonctionnelle (LU2IN019) :

    `http://www-licence.ufr-info-p6.jussieu.fr/lmd/licence/2019/ue/LU2IN019-2019oct/`
  - modèles de programmation et interopérabilité des langages (LU3IN008) :

    `http://www-licence.ufr-info-p6.jussieu.fr/lmd/licence/2018/ue/3I008-2019fev/`
- MOOC OCAML :

  `https://www.fun-mooc.fr/courses/course-v1:parisdiderot+56002+session04/about`
- autres références sur : `https://ocaml.org/learn/books.html`,

  `http://ocaml.org/docs/cheat_sheets.html`,