

Mise à Niveau et Ouverture - STL

Cours 1

Emmanuel Chailloux

Parcours Science et Technologie du Logiciel
Master mention Informatique
Sorbonne Université

année 2019-2020

Informations sur le cours d'Ouverture

Sites

https:

[//www-apr.lip6.fr/~chaillo/Public/enseignement/2019-2020/ouv](https://www-apr.lip6.fr/~chaillo/Public/enseignement/2019-2020/ouv)

<https://www-apr.lip6.fr/~genitrini/>

Equipe pédagogique

- ▶ cours : **Emmanuel Chailloux (LIP6), Antoine Genitrini (LIP6)**
- ▶ TME : **Emmanuel Chailloux (LIP6), Antoine Genitrini (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 probabilistes
 - ▶ 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 17.9 au 15.10) 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 = 60% réalisé en OCaml
 - ▶ soutenances les 19 et 26 novembre 2019
- ▶ 5 conférences :
 - ▶ mardi de 14h-16h
 - ▶ du 19.11 au 17.12
 - ▶ remise d'une synthèse d'une des conférence = 20%
 - ▶ présence obligatoire = 20%

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

```
1 # let rec map(f,l) = match l with
2   [] -> []
3   | t::q -> let nt = f(t) in
4             let nq = map(f,q) in
5             nt :: nq ;;
6 val map : ('a -> 'b) * 'a list -> 'b list = <fun>
7
8 # let compose(f,g,x) = f(g(x));;
9 val compose : ('a -> 'b) * ('c -> 'a) * 'c -> 'b = <fun>
10
11 # let addNL o = o#toString() ^ "\n";;
12 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

```
1 # type couleur = Pique | Coeur | Carreau | Trefle;;
2 # type carte = As of couleur
3     | Roi of couleur
4     | Dame of couleur
5     | Valet of couleur
6     | Autre of couleur * int ;;
7 # let valeur couleur_atout cr = match cr with
8 | As _ -> 11
9 | Roi _ -> 4
10 | Dame _ -> 3
11 | Valet c -> if c = couleur_atout then 20 else 2
12 | Autre (_,10) -> 10
13 | Autre (c,9) -> if c = couleur_atout then 14 else 0
14 | _ -> 0 ;;
15 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>
démarre le 22.9.19
- ▶ autres références sur : <https://ocaml.org/learn/books.html>,
http://ocaml.org/docs/cheat_sheets.html,