Programmation fonctionnelle

TP 4

Giron David thor@epitech.net

Résumé: Ce 4ème TP de programmation fonctionelle avec OCaml vous familiarisera avec les traits impératifs de ce langage. Certaines parties vous sont nécessaires pour le second projet `Bistromathique (Reloaded!)`*, notamment les références et les entrées/sorties.*

KOALA

Table des matières

Chapitre I

Programmation impérative

Nous avons découvert pendant le cours qu'`OCaml` proposait également une approche impérative. La programmation fonctionelle permet un environnement constant puisqu'on utilise la pile pour faire évoluer les valeurs manipulées. La programmation impérative, elle, au contraire, fait évaluer les valeurs manipulées en changeant leur état en mémoire. On parle alors de variables. Le concept de variable, au sens du `C` par exemple, n'existe pas en `Ocaml`. Pour les simuler, on utilisera des références.

Comme précisé pendant le cours, OCaml propose une syntaxe pour les boucles `while` et les boucles `for`. Bien que leur utilisation vous soit interdite dans les projets du module, il est bon pour votre culture de les connaître.

Je vous propose donc une petite série d'exercices pour faire le point sur vos connaissances et maitrises respectives des styles impératifs et fonctionnels.

I.1 Exercice 1

- Réécrivez la fonction `print_upto : int -> unit` suivante, de manière récursive (terminale) :

```
let print_upto n =
  for i = 1 to n do
    print_int i
  done;;
```

Vous savez maintenant transformer une boucle (`for` ou `while`) en récursion.

I.2 Exercice 2

- Écrivez ou déterrez la fonction :
  `func_fact : int -> int`

qui calcule la factorielle de son paramètre. Cette fonction doit utiliser uniquement la récursivité terminale pour calculer son resultat.

- Réflechissez sur l'évolution des valeurs à chaque appel récursif.

I.3 Exercice 3

- Écrivez la fonction :
  `imp_fact : int -> int`
  qui calcule la factorielle de son paramètre. Cette fonction doit utiliser uniquement des boucles pour calculer son résultat.
- Réflechissez sur l'évolution des valeurs à chaque tour de boucle.
- Comparez avec l'exercice précédent.

I.4 Exercice 4

Avec les notions du cours, il est tout à fait possible d'écrire une liste chaînée en OCaml telle que vous les écrivez en C. La preuve : vous allez le faire.

- Lisez la documentation du type `OCaml "option"` à l'adresse http://ocaml-lib.sourceforge.net/doc/Option.html.
- Ecrivez un module `ChainedList` implémentant l'interface suivante :

```
1          type 'a t
2
3          val new_list : unit -> 'a t
4          val length : 'a t -> int
5          val add : 'a t -> 'a -> unit
6          val iter : 'a t -> ('a -> unit) -> unit
```

- La définition du type `'a t` sera le record suivant :
  `{data : (('a elem) option) ref; len : int ref}`
  exporté de façon abstraite par la signature du module
- Le type `'a elem` dont dépend le type `'a t` défini par le record suivant :
  `{content : 'a; next : (('a elem) option) ref}`
  (local au module car absent de l'interface).
- Une fonction locale
  `new_elem : 'a -> elem`
  créant une nouvelle valeur de type `elem` avec ses champs `content` et `next` initialisés par défaut peut se montrer très pratique.

Chapitre II

Les entrées/sorties simples

Nous avons vu dans le cours la notion de `channel` et comment lire ou écrire dessus. Votre projet `Bistro (Reloaded!)` devant être capable de lire son entrée depuis l'entrée standard ou depuis un fichier, nous allons réaliser une commande `cat` sans arguments pour nous entraîner.

II.1 Exercice 5

- Lisez la section `Input/output` de la documentation du module Pervasives.
- Écrivez un programme nommé `cat` ayant un comportement similaire au binaire `cat` sans options.
- Ajoutez l'option `"-e"` à votre programme avec le comportement usuel de cette option.

II.2 Exercice 6

Nous allons maintenant travailler sur les palindromes, ces chaînes qui peuvent être lues à l'envers (par exemple « radar », « elle », ou encore, aux espaces près, « Esope reste ici et se repose », *etc.*).

- Écrivez une fonction `palindrome : string -> bool` qui renvoie `true` si la chaîne passée en argument est un palindrome (un vrai, celui d'Esope ne marchera pas) :

```
# palindrome "radar";;
- : bool = true
# palindrome "elle";;
- : bool = true
# palindrome "plop";;
- : bool = false
# palindrome "rad ar";;
- : bool = false
```

Vous pourrez utiliser l'exception `Exit` pour mettre fin au calcul dès que possible.

- À l'aide de la fonction `palindrome`, écrivez une nouvelle fonction `search_palindrome : string -> unit` qui ouvre en lecture un fichier dont le nom est passé en paramètre, et affiche à l'écran les lignes du fichier qui sont un palindrome.

- Testez cette fonction sur `/usr/share/dict/words`. Quel est le plus long palindrome en Anglais ? Et en Français ?

Chapitre III

Les tableaux

Les tableaux sont très souvent utilisés pour représenter des espaces en deux dimensions, dans un jeu vidéo comme `Bomberman` ou `Pacman` par exemple. Toutefois, il est toujours plus efficace de représenter un espace à deux dimensions par un tableau à une seule dimension en effecuant des calculs d'offsets.

III.1 Exercice 7

- Ecrivez un module `Grid` implémentant l'interface suivante :

```
1         type cell = {x : int; y : int; mutable content : string}
2         type t
3
4         val new_grid : int -> int -> t
5         val get_cell : t -> int -> int -> cell
```

- La définition du type `Grid.t` devra être un record contenant un tableau de `cell` à une seule dimension, la largeur et la hauteur de la grille.
- Pour vous aider dans la définition de la fonction `get_cell`, écrivez une fonction locale au module `Grid` `index_of_coords : t -> int -> int -> int` qui calcule l'index dans un tableau à une seule dimension d'une case à partir de ses coordonnées.

Bon courage !

Chapitre IV

Conclusion

Nous espérons que vous avez apprécié ce sujet autant que nous avons apprécié le rédiger pour vous.

Vos avis sont très importants pour nous et nous permettent chaque jour d'améliorer nos contenus. C'est pourquoi nous comptons beaucoup sur vous pour nous apporter vos retours.

Si vous trouvez que certains points du sujet sont obscurs, pas assez bien expliqués ou tout simplement contiennent des fautes d'orthographe, signalez-le nous. Pour cela, il vous suffit de nous envoyer un mail à l'adresse koala@epitech.eu.

Pour aller plus loin dans votre apprentissage de la programmation fonctionnelle, nous vous conseillons le livre "Développement d'applications avec Objective CAML" : http://bibliotech.epitech.eu/?page=book&id=156 ou sa version en anglais dont le PDF est disponible : http://bibliotech.epitech.eu/?page=book&id=155.

Il existe bien sûr d'autres livres sur la programmation fonctionnelle : http://bibliotech.epitech.eu/?page=search&categ=15. N'hésitez pas à vous renseigner auprès des koalas, ils seront ravis de vous conseiller. Certains livres sont disponibles en version PDF si vous êtes connecté sur le site.