

PROGRAMMATION FONCTIONNELLE

PLIAGES DE DONNÉES

L'objectif de cette dernière séance de TDs est de travailler la notion de *pliage* de données. Le pliage est une technique spécifique à la programmation fonctionnelle et qui généralise, de manière plus propre, plus complète et souvent plus rapide les boucles `for` des langages de programmation.

Exercice 1. Sans utiliser `for`, écrire une fonction `pour`, qui servira à appliquer successivement une fonction à un intervalle d'entiers, en partant d'une valeur initiale et en réutilisant à chaque fois le résultat précédent. Cette fonction constitue un *pliage sur un intervalle d'entiers*.

On aura

```
# pour ;;
- : init:'a -> de:int -> a:int -> appliquer:( 'a -> int -> 'a ) -> 'a = <fun>
```

où `init` représente la valeur initiale, `de` représente l'entier où commencer à compter, `a` représente l'entier où s'arrêter de compter et `appliquer` est la fonction à appliquer. En d'autres termes, `pour ~init:i ~d:d ~a:a ~appliquer:f` aura pour résultat `f a (f (a-1) (f (a-2) (... f (d i) ...)))`.

Exemple :

```
# let ajoute i j = i + j;;
val ajoute : int -> int -> int = <fun>

# pour ~init:0 ~de:3 ~a:5 ~appliquer:ajoute;;
- : int = 12
```

Essentiellement, cette instruction aura donc effectué l'opération

```
ajoute (5 (ajoute 4 (ajoute 3 0) ) )
```

Exercice 2. À l'aide de `pour`, définissez une fonction qui servira à appliquer successivement une fonction à tout un tableau, en partant d'une valeur initiale et en réutilisant à chaque fois le résultat précédent. Cette fonction constitue un *pliage sur un tableau*.

Exemple :

```
# pour_tableau ;;
- : 'a array -> init:'b -> appliquer:(int -> 'b -> 'a -> 'b) -> 'b = <fun>
```

```
(**
Ajoute un caractère à la fin d'une chaîne de caractères.
*)
```

```
# let ajouter_caractere _ x y = x ^ (String.make 1 y);;
val ajouter_caractere : 'a -> string -> char -> string
```

```
# pour_tableau [|'c';'o';'u';'c';'o';'u'|] ~init:""
~appliquer:ajouter_caractere;;
- : string = "coucou"
```

```
(**
Ajoute un caractère une ou plusieurs fois à la fin d'une chaîne de caractères.
```

```

*)
# let ajouter_caractere_tordu i x y = x ^ (String.make (i + 1) y);;
val ajouter_caractere_tordu : int -> string -> char -> string

# pour_tableau [|'c';'o';'u';'c';'o';'u'|] ~init:""
  ~appliquer:ajouter_caractere_tordu;;
- : string = "coouuuccccooooouuuuu"

```

Exercice 3. La fonction `ListLabels.fold_left` est un *pliage sur les listes*. À partir de cette fonction et sans définir de fonction récursive, définissez la concaténation d'une liste de caractères en une chaîne de caractères.

Exercice 4. * Apprenez à jouer à Othello/Reversi. Si, si.

Exercice 5. *** Finissez votre projet. N'oubliez pas le rapport, il compte pour un tiers des points.

Exercice 6. * Allez voter.

Exercice 7. * Révisez pour l'examen. Pour ce faire, la meilleure méthode est de refaire tous les exercices de TDs depuis le début du semestre, en particulier ceux qui n'ont pas été corrigés. N'hésitez pas à poser des questions.