Programmation Fonctionnelle: Impératif

Adrien Durier

03 octobre 2023

Boucles `for`

```
for i = 1 to 5 do
    Printf.printf "%d " i
done
```

- `i` est non modifiable et visible uniquement dans le corps de la boucle.
- `i` est incrémenté (ou décrémenté avec `downto`) à chaque tour.
- L'expression entre `do` et `done` ne doit faire que des effets de bord (warnings).
- Les bornes ne sont évaluées qu'une seule fois.

```
for i = (printf "*"; 0) to (printf "."; 5) do
  Printf.printf "%d" i
done;;
```

Affiche `*.012345`

Références `ref`

- Déclaration d'une référence de type int ref :

  ```
  let x = ref 10
  ```

  Une `int ref` est un pointeur vers une cellule mémoire contenant un `int`.

- Modification (affectation)

  ```
  x := 11
  ```

- Accès (déréférencement) :

  ```
  !x
  ```

Mutabilité

On peut modifier un champ mutable avec `<-`.

```
type student = { number : int; mutable age : int}

let birthday e = e.age <- e.age + 1

let e = { number = 12134; age = 21}

let ()= birthday e; print_int e.age
```

Affiche 22.

Références et mutabilité

Les références ne sont rien de plus que des enregistrements à un seul champ mutable. Moralement, c'est comme si les définitions suivantes avaient été effectuées :

```
type 'a ref = { mutable content : 'a }

let ref = fun v ->  { content = v }
let (:=) = fun r v -> r.contents <- v
let (!) = fun r -> r.contents
```