

Cours 3 : Tableaux et boucles

1 Tableaux

Les tableaux peuvent être créés en OCaml avec la syntaxe `[|...|]` comme dans l'exemple ci-dessous.

```
# let t = [|4;5;6|];;  
val t : int array = [|4; 5; 6|]
```

Le type des tableaux contenant des éléments de type `'a` est noté `'a array`. L'accès à la case `i` d'un tableau `t` s'écrit `t.(i)`. La mise à jour d'une case s'effectue à l'aide de l'opérateur `<-` (comme pour les champs modifiables des enregistrements).

```
# t.(0) <- 10;  
- : unit = ()  
# t.(0);;  
- : int = 10
```



L'indexation des cases d'un tableau contenant n éléments se fait de 0 à $n - 1$.

On peut aussi créer des tableaux en utilisant les fonctions prédéfinies `Array.make`, de type `int -> 'a -> 'a array`, et `Array.init` de type `int -> (int -> 'a) -> 'a array`.

```
# let t = Array.make 4 'a';;  
val t : char array = [|'a';'a';'a';'a'|]  
# let t = Array.init 5 (fun i -> 2 * i);;  
val t : int array = [|0; 2; 4; 6; 8|]
```



Un tableau créé par `Array.make n v` contient la même valeur `v` dans toutes ses cases. Attention donc aux éventuels problèmes de partage!



Comme pour les références, les tableaux sont des structures monomorphes.

```
# let t = [| [] |];;  
val t : '_a list array = [| [] |]
```

2 Boucles

On dispose de deux structures de boucle en OCaml. La boucle `for` permet d'exécuter un nombre donné de fois un bloc d'instructions délimité par les mots-clé `do` et `done`.

```
# for i = 0 to 5 do
  print_int i
done;;
- : unit = ()
012345
```

La variable de “comptage” introduite par `for` n'est visible que dans le corps de la boucle. Elle n'est pas modifiable mais est incrémentée automatiquement à chaque tour de boucle. Elle peut être décrémentée en utilisant `downto` à la place de `to`.

La boucle `while` permet de répéter un bloc d'instructions tant qu'une certaine condition est vérifiée.

```
# let i = ref 0 in
  while !i <= 5 do
    print_int !i;
    incr i
  done;;
- : unit = ()
012345
```

Dans ces deux boucles, le bloc d'instructions doit être de type `unit` (dans le cas contraire le compilateur émet un *warning*).



Dans `for i=e1 to e2 do ... done`, les expressions `e1` et `e2` sont évaluées une seule fois à l'initialisation de la boucle.