

Correction de devoir du Vendredi 28 Février

Exercice 1 (La syntaxe en OCaml)

1. Le type `array` est un type complexe ayant les caractéristiques suivantes :
 - les objets le composant sont tous de même type,
 - chaque objet est directement accessible (accès en temps constant),
 - l'ensemble est de taille fixe (type statique),
 - la valeur de chaque objet peut être modifiée (type mutable).
 Le type `list` est un type complexe ayant les caractéristiques suivantes :
 - les objets le composant sont tous de même type,
 - chaque objet contient l'adresse du suivant et ainsi le temps d'accès à chaque objet n'est plus identique (accès en temps linéaire),
 - l'ensemble est de taille modifiable (type dynamique),
 - la valeur de chaque objet ne peut être modifiée (type non mutable).
2. (a) - : `bool = true`
 (b) - : `int = 0`
 (c) - : `'a list list = [[]]`
 (d) Erreur car 2 est de type `int` et devrait être de type `float`.
 (e) - : `(int * int) list = [(1, 2); (3, 0)]`
 (f) - : `char = 'f'`
 (g) - : `int array list = [[|2;3|]; [|1;2;4|]]`
 (h) - : `'a array -> int -> int -> 'a array = <fun>`
 (i) - : `int array = [|2; 4|]`
 (j) Erreur car `[1,2;4,5]` n'est pas un tableau.
3. (a) `(1. +. sqrt(5.)) /. 2.;;`
 - : `float = 1.6180339887498949`
 (b) `[|1;4;5;2;8|].(2) <- 0;;`
 - : `unit = ()`
 (c) `List.hd (List.tl (List.tl [|1;4;5;2;8|]));;`
 - : `int = 5`
 (d) `Array.make 3 2;;`
 - : `int array [|2; 2; 2|]`
4. (a) `x` est de type `int list list` et `List.length ([2;4]::x) ;;` est de type `int`.
 (b) `x` est de type `int list list` et `x@[|1|]` ;; est aussi de type `int list list`.
 (c) `x` est de type `'a` et `x::[]` ;; est de type `'a list` (il y a un polymorphisme de type ici).

Exercice 2 (Du cours...)

1. Factorielle :

(a)

```
let fact_iter n =
  let f = ref 1 in
  for k = 1 to n do
    f := !f * k
  done;
  !f
;;
```

(b)

```

let rec fact_rec n =
  if (n = 0)
  then
    1
  else
    n * (fact_rec (n-1))
;;

```

2. Suite récurrente :

(a)

```

let suite_iter n =
  let u = ref 2. in
  for k = 1 to n do
    u := log (1. +. !u)
  done;
  !u
;;

```

(b)

```

let rec suite_rec n =
  if (n = 0)
  then
    2.
  else
    log (1. +. (suite_rec (n-1)))
;;

```

Exercice 3 (Manipulation récursive de tableau en place)

1.

```

let maximum a =
  let p = ref 0 in
  for i = 1 to (Array.length a - 1) do
    if a.(i) > a.(!p) then
      p := i
  done;
  !p
;;

```

2. (a) • Si $i = j$, alors, il n'y a qu'une seule valeur dans le tableau donc $p = i$.

- Si $i < j$, soit un indice q de la valeur maximale parmi les valeurs $a[k]$, k appartenant à $\llbracket i + 1, j \rrbracket$, c'est-à-dire tel que :

$$a[q] = \max(\{a[i + 1], a[i + 2], \dots, a[j - 1], a[j]\}).$$

On compare alors $a[i]$ avec $a[q]$: si $a[i] > a[q]$ alors l'indice p tel que

$$a[p] = \max(\{a[i], a[i+1], \dots, a[j-1], a[j]\}).$$

est $p = i$, sinon, le maximum est atteint en $p = q$.

- ```
let rec maximum_aux a i j=
 if (i=j)
 then
 i
 else
 let q = (maximum_aux a (i+1) j) in
 if a.(i) > a.(q)
 then
 i
 else
 q
;;
```

(b)

```
let maximum a =
 let n=Array.length a in maximum_aux a 0 (n-1)
;;
```

---