TP : Programmation impérative et Tableaux

Si OCaml est un langage fonctionnel, on peut quand même l'utiliser pour de la programmation impérative.

Une liste d'instructions OCaml est séparée par des points-virgules.

On note qu'une fonction Caml renvoie nécessairement une valeur. Dans le cas où on ne veut rien renvoyer, on peut utiliser le type `unit` ; il contient une unique valeur `():unit`.

EXEMPLE

> La fonction `print_int` est de type `int -> unit` : elle prend un entier et l'affiche, mais ne renvoie pas de valeur.

1 Boucles

Une boucle inconditionnelle s'écrit en OCaml

```
for i=... to ... do
   instructions
done;
```

On peut aussi faire tourner la boucle dans l'autre sens :

```
for i=... downto ... do
   instructions
done
```

Les boucles conditionnelles s'écrivent quand à elles

```
while condition do
  instructions
done
```

2 Références

En programmation impérative, on a un accès direct à la mémoire, là où c'est impossible en programmation purement fonctionnelle. Une *référence* est un objet Caml qui pointe vers une adresse mémoire. À cette adresse mémoire se trouve une valeur.

Attention à ne pas confondre la référence et la valeur contenue à l'adresse de la référence.

En OCaml, on définit une référence avec `let nom = ref valeur`. Dans ce cas, la valeur `valeur` peut être obtenue avec la commande `!nom`. On peut changer la valeur de la référence avec la commande `nom := nouvelle_valeur`.

La différence entre une affectation `let nom = valeur` et la définition de la référence est le caractère statique du premier cas : on pourra essayer les programmes suivants

```
# let a = 1;;
# let succ n = n + a;;
# let a = 2;;
# succ 0;;
```

```
# let a = ref 1;;
# let succ n = n + !a;;
# a := 2;;
# succ 0;;
```

3 Tableaux

Les tableaux, ou vecteurs, en OCaml sont des objets très utilisés en programmation impérative. Le type des tableaux d'objets de type `'a` est noté `'a array`.

Il y a deux façons de créer un tableau : directement en donnant ses éléments, avec la syntaxe `[|v1;v2;...;vn|]`, ou en utilisant la fonction `Array.make n v` qui crée un tableau à `n` cases contenant toutes la valeurs `v`.

Les tableaux Caml ne sont pas redimensionnables ; le nombre d'éléments d'un tableau est fixé à sa création, et ne peut être changé. En revanche, on peut changer ses éléments : si `tab:'a array` et `a:'a`, alors la commande `tab.(i) <- a` change la i-ième case (si elle existe) du tableau par la valeur i. Cette instruction est de type `unit`.

L'accès aux éléments du tableau se fait avec `tab.(i)`, où `i` est le numéro de la case voulue (attention, elles sont numérotées à partir de 0).

On renvoie à la documentation d'OCaml pour connaître les fonctions usuelles sur les tableaux, en particulier :

- `Array.length : 'a array -> int` donne la taille d'un tableau
- `Array.make : int -> 'a -> 'a array` crée un tableau constant
- `Array.make : int -> int -> 'a -> 'a array array` crée une matrice constante
- `Array.mem : 'a -> 'a array -> bool` vérifie l'appartenance d'un élément à un tableau

- `Array.sub : 'a array -> int -> int -> 'a array` crée un sous-tableau ; plus précisemment, `Array.sub tab start len` crée un tableau de longueur `len` contenant les éléments `tab.(start)` à `tab.(start+len-1)`.

4 Exercices

Exercice 1. Définir la fonction `factorielle : int -> int` de façon impérative.

Exercice 2. Écrire une fonction calculant le maximum d'un tableau ; on donnera une version impérative et une version fonctionnelle.

Exercice 3. Écrire une fonction `swap : 'a array -> int -> int -> unit` qui échange deux éléments d'un tableau.

Exercice 4. Écrire une fonction `table_mult : int -> 'a array array` qui renvoie la matrice des tables de multiplication de 1 à l'entier choisi. Par exemple, `table 3` doit renvoyer le tableau

$$\begin{matrix} 1 & 2 & 3 \\ 2 & 4 & 6 \\ 3 & 6 & 9 \end{matrix}$$

Exercice 5. Écrire une fonction qui vérifie si un tableau est un palindrome.

Exercice 6. Écrire une fonction impérative qui calcule le miroir d'une liste.

Exercice 7. Écrire une fonction `dichotomie : (float -> float) -> float -> float -> float` prenant un paramètre une fonction $f : \mathbb{R} \to \mathbb{R}$ et deux réels $a < b$ tels que $\exists c \in]a, b[,\ f(c) = 0$, et qui calcule une approximation de x à 10^{-7} par dichotomie.

Exercice 8. Écrire une fonction qui implémente la produit (naïf) de matrices.