TP : Exploration des traits impératifs en OCaml

Introduction

Dans ce TP, nous allons explorer les traits impératifs en OCaml à travers des exercices portant sur les tableaux et les enregistrements. Pour chaque exercice, la signature de la fonction à implémenter est donnée dans une commande `_`. Les exercices sont regroupés en deux parties.

1 Manipulation de tableaux

Les exercices suivants portent sur la manipulation des tableaux en OCaml.

Exercice Recherche du minimum.

Objectif : Écrire une fonction qui prend en argument un tableau de valeurs et retourne le minimum de ce tableau.
Signature : `min_tab : 'a array -> 'a`
Remarque : Vous pouvez supposer que le tableau n'est pas vide.

Exercice Inversion de tableau.

Objectif : Écrire une fonction qui inverse les éléments d'un tableau *in-place*.
Signature : `reverse_tab : 'a array -> unit`

Exercice Transposée de matrice.

Objectif : Écrire une fonction qui calcule la transposée d'une matrice représentée par un tableau de tableaux.
Signature : `transpose : 'a array array -> 'a array array`

Exercice Drapeau hollandais.

Objectif : Écrire une fonction qui réarrange un tableau contenant les entiers 0, 1 et 2 de manière à les ordonner en un seul parcours (exercice du drapeau hollandais).
Signature : `dutch_flag : int array -> unit`

Exercice Tri à bulle.

Objectif : Écrire une fonction qui trie un tableau en utilisant l'algorithme du tri à bulle.
Signature : `bubble_sort : 'a array -> unit`
Description : Le tri à bulle, ou *bubble sort*, est un algorithme de tri simple qui fonctionne de la manière suivante :

1. Le tableau est parcouru en comparant successivement chaque paire d'éléments adjacents.
2. Si deux éléments sont dans le mauvais ordre (pour un tri croissant, si l'élément de gauche est supérieur à l'élément de droite), ils sont échangés.
3. Ce processus est répété pour l'ensemble du tableau plusieurs fois. À chaque passage, l'élément le plus grand "bulle" vers la fin du tableau et se retrouve à sa position finale.

4. Lorsque plus aucun échange n'est nécessaire, le tableau est considéré comme trié.

Remarque : Bien que simple à implémenter, cet algorithme a une complexité en $O(n^2)$ dans le pire des cas, ce qui le rend inefficace pour trier de grands tableaux.

2 Manipulation d'enregistrements

Les exercices suivants portent sur l'utilisation des enregistrements en OCaml. Vous utiliserez notamment les enregistrements pour modéliser des nombres complexes.

Exercice Création d'un enregistrement complexe.

Objectif : Définir un type `complexe` pour représenter un nombre complexe, avec des champs pour la partie réelle et la partie imaginaire. Écrire une fonction qui crée un complexe à partir de deux réels.
Signature : `make_complex : float -> float -> complexe`

Exercice .

Écrire une fonction `affiche_complexe : complexe -> unit` qui affiche un complexe.

```
# affiche_complexe (make_complex 2.36 3.8956);;
(2.36, 3.90)- : unit = ()
```

Exercice Addition de nombres complexes.

Objectif : Écrire une fonction qui additionne deux nombres complexes.
Signature : `add_complex : complexe -> complexe -> complexe`

Exercice Multiplication de nombres complexes.

Objectif : Écrire une fonction qui multiplie deux nombres complexes.
Signature : `mul_complex : complexe -> complexe -> complexe`

Exercice Conjugué.

Objectif : Écrire une fonction qui donne le conjugué d'un complexe
Signature : `conjugue : complexe -> complexe`

Exercice .

Écrire une fonction `print_complexe_array : complexe array -> unit` qui affiche le contenu d'un tableau de complexes.

```
# print_complexe_array [|make_complex 2. 6.; make_complex 0. 1.;
    make_complex 8.36 5.365|];;
(2.00, 6.00) (0.00, 1.00) (8.36, 5.37)
- : unit = ()
```

Exercice Module.

Objectif : Écrire une fonction qui donne le module d'un complexe
Signature : `module : complexe -> complexe`

Exercice Tri à bulle (bis).

Écrire une fonction `tri_bulle : complexe array -> unit` qui réalise le tri par module croissant d'un tableau de complexes.

Exercice Tri par insertion.

Écrire une fonction `tri_insertion : complexe array -> unit` qui réalise le tri par module croissant d'un tableau de complexes.