TP 4 : Mutabilité en OCaml

3 décembre 2023

1 Rappels : Traits Impératifs en OCaml

*

2 Enregistrements Modifiables et Références

Exercice 1

Ecrire des fonctions `conjugue` et `carre` qui prennent en entrée un complexe mutable et modifient sa valeur pour la remplacer par son conjugué et son carré respectivement.

```
let conjugue z =
    z.im <- -.z.im ;;
```

```
let carre z =
    let re1 = z.re *. z.re -. z.im *. z.im
    and im1 = z.re *. z.im *. 2. in
    z.re <- re1; z.im <- im1 ;;
```

2.1 Références

*

2.2 Egalité physique et structurelle

*

Exercice 4

Ecrire une fonction `affichage_syracuse` qui prend en entrée un entier n et un seuil s et affiche la liste des termes de la suite de Syracuse à partir de n avant le premier terme inférieur au seul s (le programme n'affiche aucun terme si n et inférieur à s).

```
let affichage syracuse n s =
    let i = ref n in
    while !i >=s do
        print_int i;
        if !i mod 2 = 0 then
        i := !i / 2;
```

```
        else
        i := 3* !i + 1;
    done
    ;;
```

3 Tableaux

*

Exercice 5

Ecrire une fonction `maximum t` qui prend en entrée un tableau d'entiers et calcule sa valeur maximum.

```
let maximum t =
    let i = ref t.(0) in
    for k = 1 to Array.length(t)-1 do
        if !i < t.(k) then i := t.(k);
    done;
    !i ;;
```

Exercice 6

Ecrire une fonction `swap t i j` qui intervertit les éléments d'indices i et j dans le tableau t.

```
let swap t i j =
    let a = t.(i) in
    t.(i) <- t.(j);
    t.(j) <- a ;;
```

Exercice 7

Ecrire une fonction `somme t` qui calcule la somme de tous les éléments de t.

```
let somme t =
    let i = ref 0 in
    for k = 0 to Array.length(t)-1 do
        i := !i + k;
    done;
    !i ;;
```

Exercice 8

Ecrire une fonction `trie` qui teste si un tableau est trié.

```
let trie t =
    let i = ref true in
    for k = 0 to Array.length(t)-2 do
        if t.(k) > t.(k+1) then i := false;
    done;
    !i ;;
```

Exercice 9

Ecrire une fonction `array_to_list t` qui transforme un tableau en liste.

```
let array_to_list t =
    let n = Array.length(t) in
    let l = ref [] in
    for k = 1 to n do
        l := t.(n-k)::(!l);
    done;
    !l ;;
```

Exercice 10

Ecrire une fonction **pascal n** qui renvoie les lignes 0 à n du triangle de Pascal (sous la forme d'un tableau de tableaux).

```
let pascal n =
  let t1 = Array.init (n+1) (fun x -> Array.make (x+1) 1) in
  for i = 2 to n do
    for j = 1 to (i-1) do
      t1.(i).(j) <- t1.(i-1).(j) + t1.(i-1).(j-1);
    done;
  done;
  t1 ;;
```

3.1 Matrices

*

Exercice 11

Ecrire une fonction `trace m` qui prend en entrée une matrice et calcule sa trace (somme des termes diagonaux).

```
let trace m =
    let a = ref 0 in
    for k = 0 to (Array.length m - 1) do
        a := !a + m.(k).(k);
    done;
    !a;;
```

Exercice 12

Ecrire une fonction `addmat m1 m2` qui prend en entrée deux matrices de mêmes tailles et calcule leur somme.

```
let addmat m1 m2 =
    let m = Array.make_matrix (Array.length m1) (Array.length (m1.(0)) ) 0 in
    for i = 0 to (Array.length m1 - 1) do
        for j = 0 to (Array.length (m1.(0)) - 1) do
            m.(i).(j) <- m1.(i).(j) + m2.(i).(j);
        done;
    done;
    m;;
```

Exercice 13

Ecrire une fonction `transpose m` qui transpose une matrice m donnée en entrée.

```
let transpose m =
    let n = (Array.length m) in
    for i = 0 to n-2 do
        for j = i+1 to n-1 do
            let a = m.(i).(j) in
            m.(i).(j) <- m.(j).(i);
            m.(j).(i) <- a;
        done;
    done;
    m;;
```

Exercice 14

Ecrire une fonction `symetrique m` qui prend en entrée une matrice carrée `m` et vérifie qu'elle est symétrique.

```
let symetrique m =
    let a = ref true in
    let n = (Array.length m) in
    for i = 0 to n-2 do
        for j = i+1 to n-1 do
            if m.(i).(j) <> m.(j).(i) then a := false;
        done;
    done;
    !a;;
```

Exercice 15

Ecrire une fonction `multmat m1 m2` qui prend en entrée deux matrices de mêmes tailles et calcule leur produit.

On rappelle que pour deux matrices A et B de tailles $n \times n$, la matrice $C = A \times B$ est la matrice dont la coordonnée à la i-ème ligne et j-ème colonne vaut : $c_{i,j} = \sum_{k=1}^{n} a_{i,k} \times b_{k,j}$.

```
let multmat m1 m2 =

  let n = (Array.length m1) in
  let m = Array.make_matrix n n 0 in
  for i = 0 to (n - 1) do
    for j = 0 to (n - 1) do
      let a = ref 0 in
      for k = 0 to (n - 1) do
        a := !a + m1.(i).(k) * m2.(k).(j);
      done;
      m.(i).(j) <- !a;
    done;
  done;
  m;;
```