Programmation fonctionnelle et Parallélisme

Cours 4

Jean-Jacques Lévy

jean-jacques.levy@inria.fr

http://jeanjacqueslevy.net/prog-fp

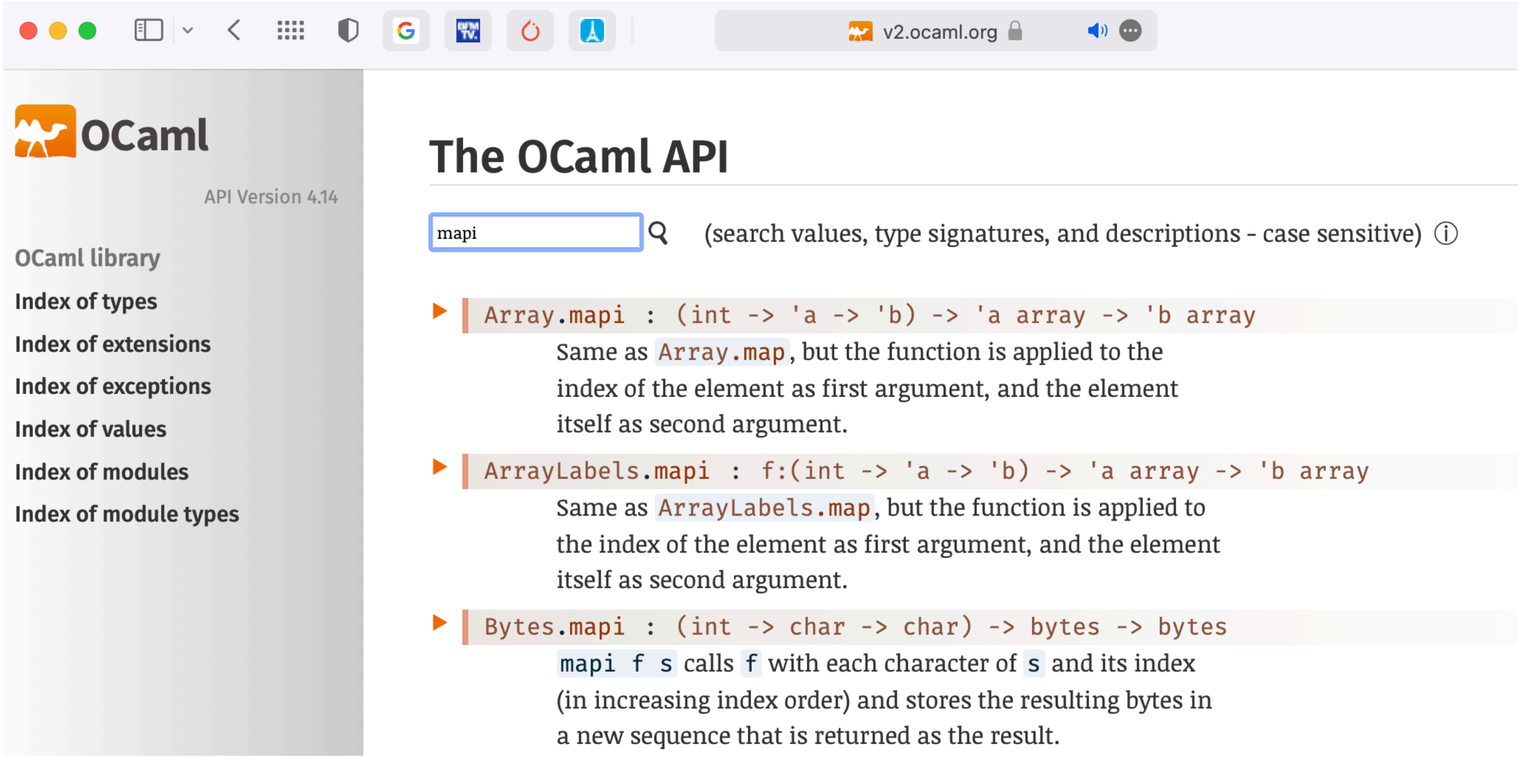
Plan

- rappels et solutions des exercices
- alias et références
- n-uplets et chaînes de caractères
- tris élémentaires
- programmes graphiques

télécharger Ocaml en `http://www.ocaml.org`

Librairie standard

- l'API de Ocaml est visible en `http://v2.ocaml.org/api`
- avec les fonctions de la librairie standard aussi en `http://v2.ocaml.org/manual/stdlib.html`



Rappels et exercices

VU:

- int, float, char, strings, array
- itérateurs sur tableaux et chaînes
- tableaux multidimensionnels
- fonctions anonymes
- types polymorphes
- exceptions

Exercice 1 Trouver l'indice du maximum dans un tableau d'entiers

Exercice 2 Trouver l'indice du premier nombre négatif dans un tableau d'entiers

Exercice 3 Trouver l'indice du dernier nombre négatif dans un tableau d'entiers

Exercice 4 Trouver l'indice du premier caractère différent dans 2 chaînes `s` et `s'` (-1 si les mêmes chaînes)

[indication: utiliser la fonction `String.iteri`]

Exceptions

- utiliser des exceptions pour rompre une évaluation
- exemple: trouver l'indice d'un élément de valeur **v** dans le tableau *a* (ou -1 si **v** n'est pas dans *a*)

```
let index_in v a =
  let exception Found of int in
  try
    Array.iteri (fun i x -> if x = v then raise (Found i)) a;
    -1
  with Found i -> i ;;
```

on lève l'exception si on a trouvé **v**

- on déclare une exception `Found` qui a un paramètre entier (`int`)
- et on lève l'exception que l'on peut récupérer avec `try` .. `with`

les sources sont en `http://jeanjacqueslevy.net/prog-fp/progs/fp4a.ml`

Exercices

Exercice 1 Trouver l'indice du maximum dans un tableau d'entiers

```
let min_in_array a = Array.fold_left min max_int a ;;

let index_min_of a = index_in (min_in_array a) a ;;
```

Exercice 2 Trouver l'indice du premier nombre négatif dans un tableau d'entiers

```
let index_fst_neg_in a =
  let exception Found of int in
  try
    Array.iteri (fun i x -> if x < 0 then raise (Found i)) a;
    -1
  with Found i -> i ;;
```

Exercice 3 Trouver l'indice du dernier nombre négatif dans un tableau d'entiers

```
let index_lst_neg_in a =
  let n = Array.length a in
  n-1 - index_fst_neg_in (miroir a) ;;
```

les sources sont en http://jeanjacqueslevy.net/prog-fp/progs/fp4a.ml

Exercices

Exercice 4 Trouver l'indice du premier caractère différent dans 2 chaînes `s` et `s'` (-1 si les mêmes chaînes)

```
let index_diff a b =
  if String.length a = String.length b then
    let exception Found of int in
    try
      String.iteri (fun i _ -> if a.[i] != b.[i] then raise (Found i)) a;
      -1
    with Found i -> i
  else -1 ;;
```

les sources sont en http://jeanjacqueslevy.net/prog-fp/progs/fp4a.ml

Exercices

Fonctions utiles:

- imprimer un tableau d'entiers

```
let print_array a =
  Array.iter (Printf.printf "%d ") a;
  print_newline() ;;
```

- générer un tableau de `n` entiers avec des valeurs aléatoires entre `0` et `p-1`

```
let rand_array n p = Array.init n (fun i -> Random.int p) ;;
```

les sources sont en `http://jeanjacqueslevy.net/prog-fp/progs/fp4a.ml`

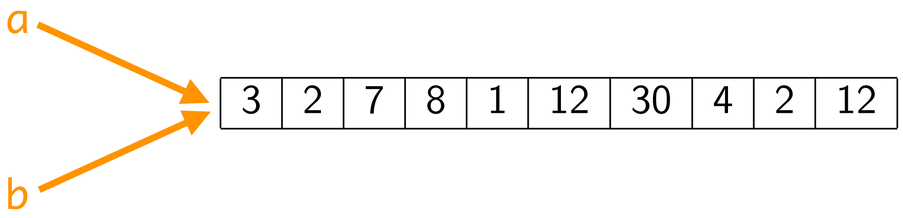
Valeur d'un tableau — Alias

- la fonction (polymorphe) suivante change la valeur d'un élément d'un tableau à un certain indice

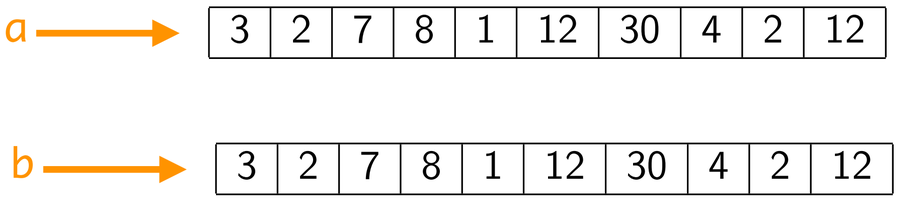
```
(* val store : int -> 'a -> 'a array -> unit = <fun> *)

let store i v a =  a.(i) <- v ;;
```

- soient 2 tableaux *a* et *b*

```
let a = [|3; 2; 7; 8; 1; 12; 30; 4; 2; 12|] ;;
let b = a ;;
store 2 888 a ;;
a ;;
- : int array = [|3; 2; 888; 8; 1; 12; 30; 4; 2; 12|]
b ;;
- : int array = [|3; 2; 888; 8; 1; 12; 30; 4; 2; 12|]
```



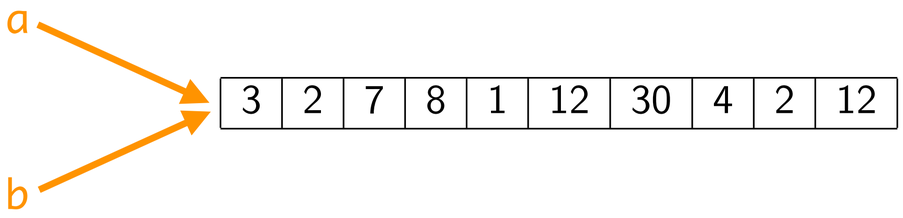

```
let a = [|3; 2; 7; 8; 1; 12; 30; 4; 2; 12|] ;;
let b = [|3; 2; 7; 8; 1; 12; 30; 4; 2; 12|] ;;
store 2 888 a ;;
a ;;
- : int array = [|3; 2; 888; 8; 1; 12; 30; 4; 2; 12|]
b ;;
- : int array = [|3; 2; 7; 8; 1; 12; 30; 4; 2; 12|]
```

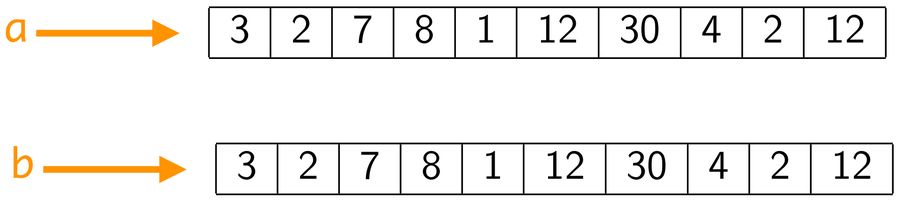


Valeur d'un tableau — Alias

- les 2 tableaux *a* et *b* sont des alias
- les valeurs de *a* et *b* sont les adresses (mémoire) où se trouvent ces tableaux
- **modification de la mémoire et alias ne font pas bon ménage**
- ce sont des sources de bugs
- la programmation fonctionnelle essaie de les éviter puisque toutes les variables sont des constantes

en Python, toutes sont modifiables !!





Références

- en Ocaml, les éléments des tableaux sont donc modifiables (contrairement à Haskell)
- mais les chaînes de caractères ne sont pas modifiables
- une référence est l'adresse d'une case mémoire modifiables

```
let r = ref 32 ;;

print_int !r ;;
```


```
r := 48 ;;

print_int !r ;;
```

Références

- une référence est identique à un tableau de 1 élément

création ⟷ `ref expression`

la valeur pointée ⟷ `!r`

affectation ⟷ `r := expression`

- la syntaxe est baroque avec les nouveaux symboles `!` et `:=`

- mais elle permet la synthèse de types

Références

- programmation impérative avec des références

```
let max_of a =
  let n = Array.length a in
  let max = ref(min_int) in
  for i = 0 to n-1 do
    if a.(i) > !max then
        max := a.(i)
  done;
  !max ;;
```

modification de la valeur

- autre exemple de programmation impérative avec des références

```
let index_max_of a =
  let n = Array.length a in
  let max = ref(min_int) and imax = ref(-1) in
  for i = 0 to n-1 do
    if a.(i) > !max then begin
        max := a.(i) ;
        imax := i
      end
  done;
  !imax ;;
```

modifications des valeurs

```
let index_max_of a =
    let n = Array.length a in
    if n = 0 then -1 else
      let imax = ref 0 in
      for i = 1 to n-1 do
        if a.(i) > a.(!imax) then
          imax := i
      done;
      !imax ;;
```

n-uplets et chaînes

- les n-uplets (*tuples*)

```
(* val fete_nationale : int * string = (14, "juillet") *)

let fete_nationale = (14, "juillet") ;;

fst fete_nationale ;;
- : int = 14

snd  fete_nationale ;;
- : string = "juillet"

(* val bastille : int * string * int = (14, "juillet", 1789) *)

let bastille = (14, "juillet", 1789) ;;
```

- itérateurs sur les chaînes de caractères

```
String.length, String.map, String.mapi, String.iter, String.iteri
```

- n-uplets et chaînes ne sont pas modifiables

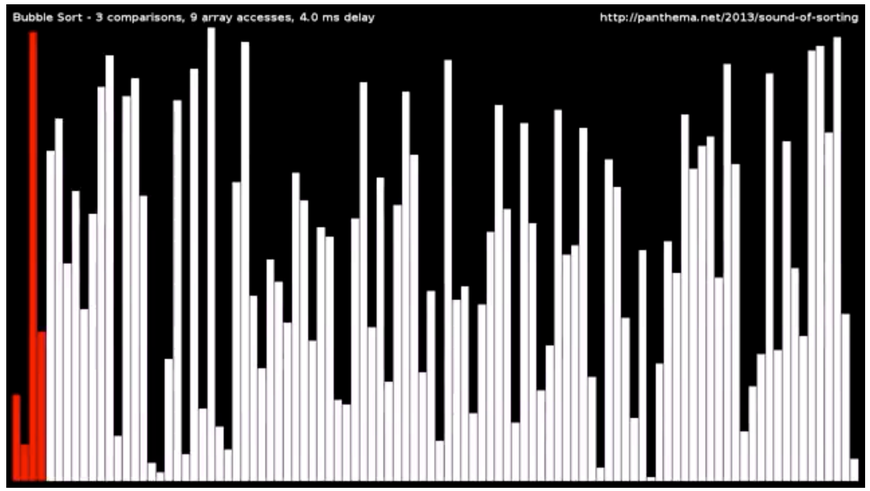
Tri bulle

- on veut trier les éléments d'un tableau d'entiers par ordre croissant

```
let bubble_sort a =
  let n = Array.length a in
  for i = n-1 downto 0 do
    for j = 0 to i-1 do
      if a.(j) > a.(j+1) then
        xchange a j (j+1)
    done
  done;;
```

```
let xchange a i j =
  let tmp = a.(i) in
  a.(i) <- a.(j);
  a.(j) <- tmp ;;
```

échanger les valeurs de `a.(i)` et `a.(j)`



les sources sont en http://jeanjacqueslevy.net/prog-fp/progs/fp4a.ml

Tri sélection

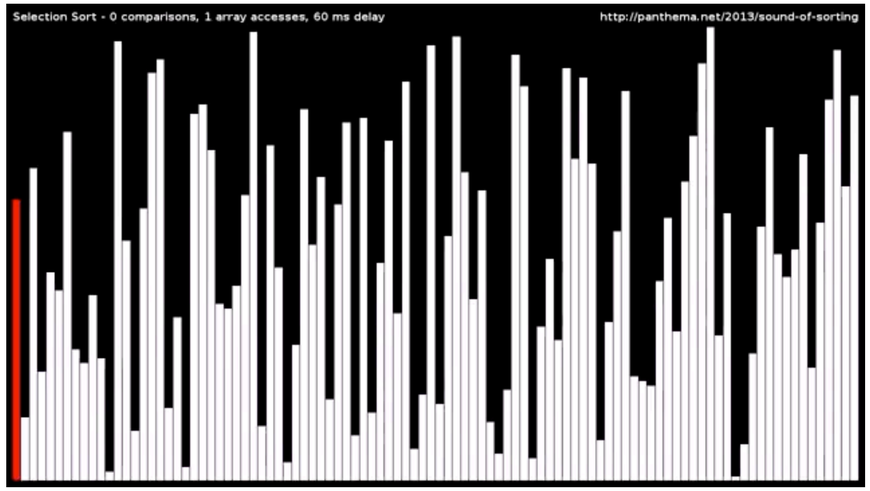
- on cherche le minimum et on le met en tête..
  et on recommence à partir du deuxième élément, etc…

```
let index_min_of_sub a i j =
  let b = Array.sub a i (j - i) in
  let k = index_in (min_in_array b) b in
  if k < 0 then k else i + k ;;
```

```
let selection_sort a =
  let n = Array.length a in
  for i = 0 to n-2 do
    let j = index_min_of_sub a i n in
    xchange a i j
  done ;;
```

```
let xchange a i j =
  let tmp = a.(i) in
  a.(i) <- a.(j);
  a.(j) <- tmp ;;
```

échanger les valeurs de `a.(i)` et `a.(j)`



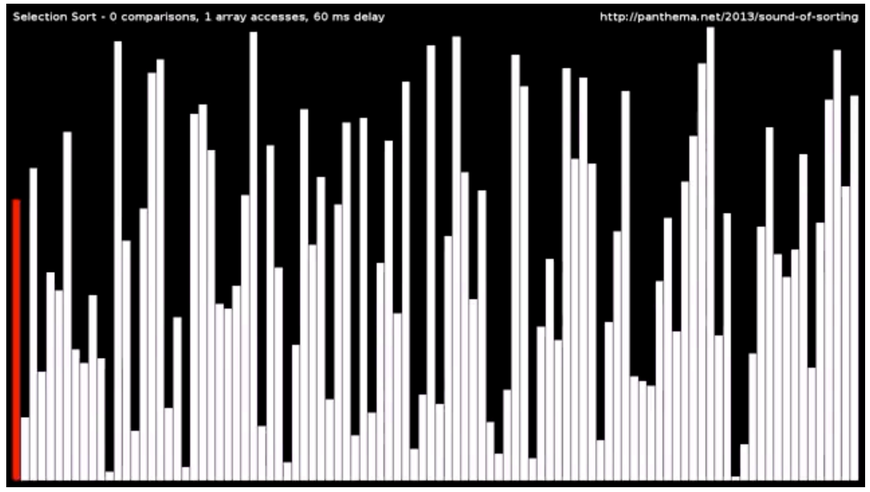
Tri sélection

- on cherche le minimum et on le met en tête..
  et on recommence à partir du deuxième élément, etc…

```
let xchange a i j =
  let tmp = a.(i) in
  a.(i) <- a.(j);
  a.(j) <- tmp ;;
```

échanger les valeurs de `a.(i)` et `a.(j)`

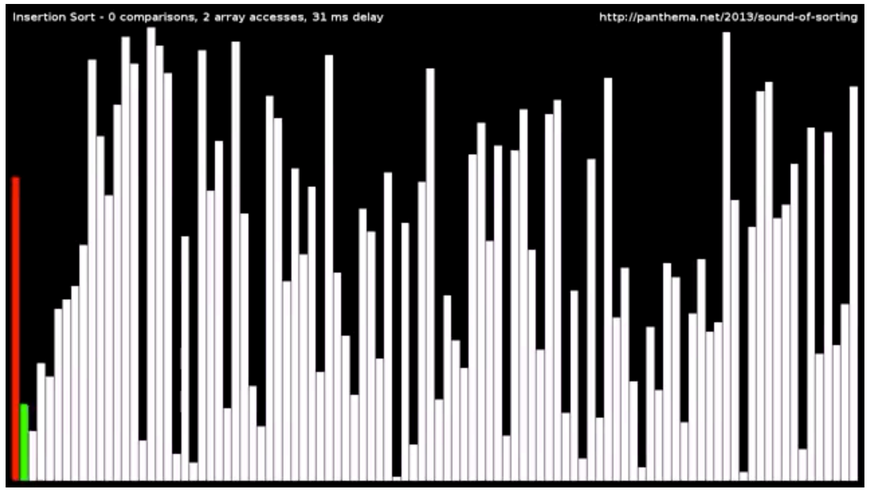
```
let selection_sort a =
  let n = Array.length a in
  for i = 0 to n-2 do
    let jmin = ref i in
    for j = i+1 to n-1 do
      if a.(j) < a.(!jmin) then
          jmin := j
    done;
    xchange a i !jmin
  done;;
```

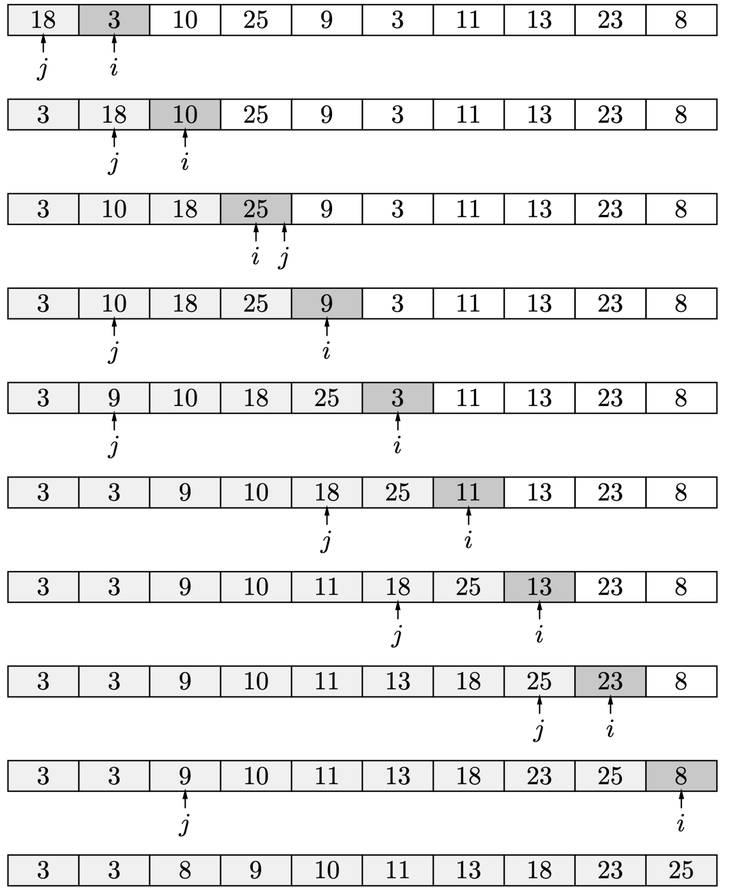


Tri par insertions

- on insère les éléments dans la partie gauche déjà triée, etc…… [comme dans un jeu de cartes]

```
let insertion_sort a =
  let n = Array.length a in
  for i = 1 to n-1 do
    let v = a.(i) and j = ref i in
    while (!j > 0) && (a.(!j-1) > v) do
      a.(!j) <- a.(!j-1);
      decr j
    done ;
    a.(!j) <- v
  done ;;
```

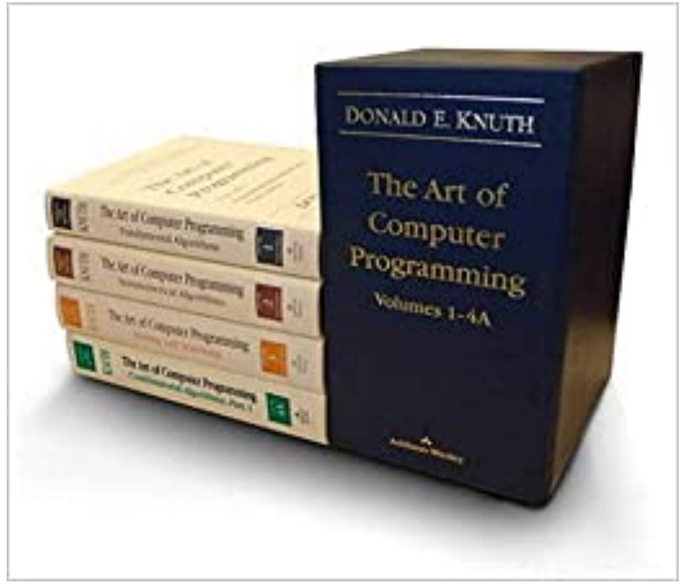


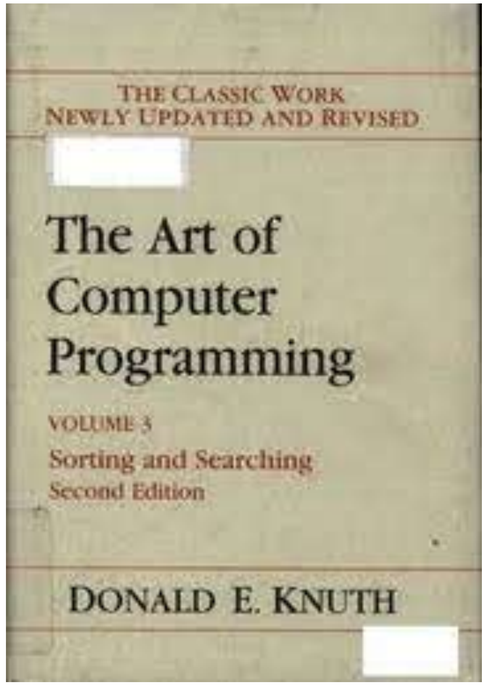


Exercices

Exercice 1 Ecrire le tri par insertions

Exercice 2 quel est le meilleur de ces 3 tris ?





Donald Knuth

Exercice 3 compter le nombre d'opérations de ces tris en fonction de la longeur n du tableau à trier

Quelques remarques

```
# selection_sort ;;
- : 'a array -> unit = <fun>
# insertion_sort ;;
- : 'a array -> unit = <fun>
# bubble_sort ;;
- : 'a array -> unit = <fun>
```

- ces fonctions de tris ont des types polymorphes

```
let a = [| 44; 127; 24; 15; 60; 149; 147; 72; 36; 34 |] ;;
let b = [| 2.3; 2.0; 4.6 |] ;;
let c = [| "camille"; "jean-jacques"; "paul"; "axel" |];;

selection_sort a  →  [|15; 24; 34; 36; 44; 60; 72; 127; 147; 149]|

selection_sort b  →  [|2.; 2.3; 4.6|]

selection_sort c  →  [|"axel"; "camille"; "jean-jacques"; "paul"|] ;;
```

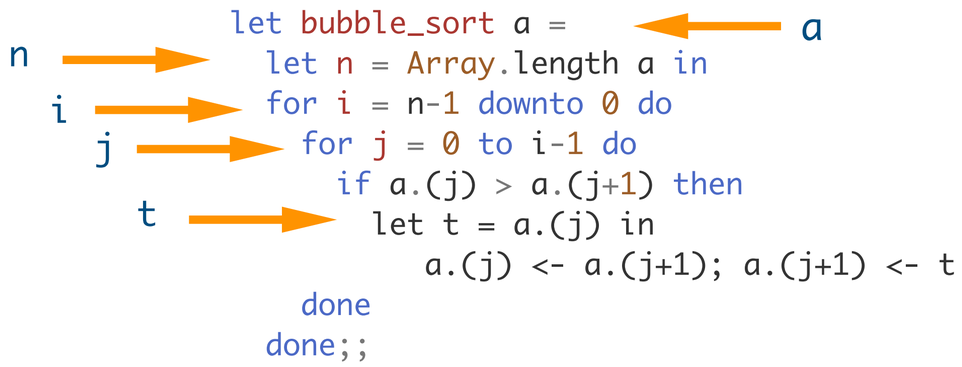
- et on voit les erreurs de types avant l'exécution

```
let d = [| "camille"; 28; "paul"; "axel"|] ;;
Error: This expression has type int but an expression was expected of type
         string

let e = [| 2.3; 2; 4.6 |] ;;
Error: This expression has type int but an expression was expected of type
         float
  Hint: Did you mean `2.'?
```

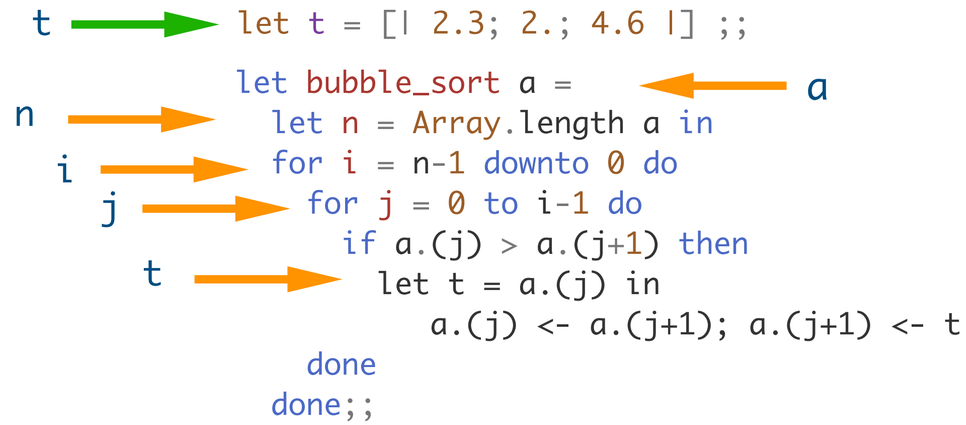
- en Ocaml, les **types sont statiques**

Quelques remarques

- les variables déclarées dans une fonction n'existent que dans le code de cette fonction



- les variables `n`, `j`, `i`, `t` et *a* sont **locales** à la fonction `tri_bulle`



- la variable `t` **globale** est distincte de la variable `t` locale

Quelques remarques

- les fonctions ou données (de librairie..) sont regroupées en modules

- notation qualifiée avec nom de module `Random.int`

```
let rand_array n p =
    Array.init n (fun i -> Random.int p) ;;
```

- on peut utiliser la notation simple `open_graph` sans le nom du module `Graphics` avec

```
open Graphics;;
```

- la liste des modules disponibles figure (en partie) dans la bibliothèque standard

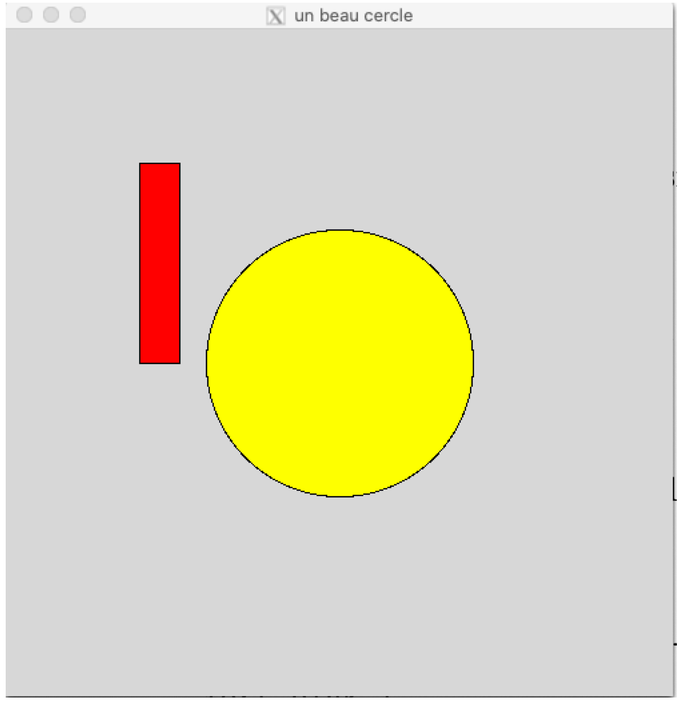
`http://v2.ocaml.org/manual/stdlib.html`

Graphique

- un paquetage `graphics.cma` simple pour dessins 2D (a besoin d'installer le module `graphics`)
  (cf. `http://v2.ocaml.org/releases/4.06/htmlman/libref/Graphics.html`)

```
#use "topfind" ;;
#require "graphics" ;;

open Graphics;;

let dessin1 () =
    open_graph "" in
    set_color yellow ;
    fill_circle 80 40 20 ;
    set_color red ;
    fill_rect 100 100 130 250 ;
    wait_next_event (Button_down :: []) ;;
```

pour éviter de taper le préfixe `Graphics.`



- pour voir les fonctions (la signature) du module Graphics

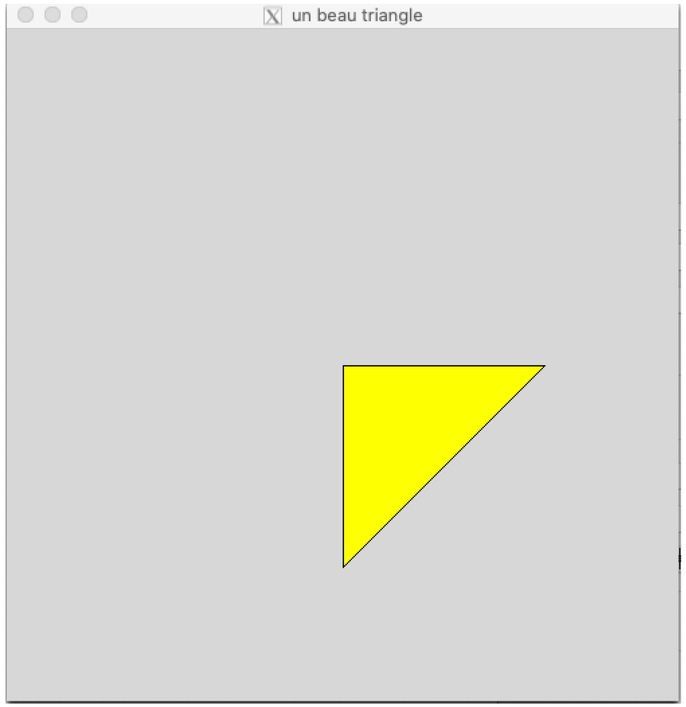
```
#show Graphics ;;
```

les sources sont en `http://jeanjacqueslevy.net/prog-fp/progs/fp4b.ml`

Graphique

- un paquetage `graphics.cma` simple pour dessins 2D (a besoin d'installer le module `graphics`)
  (cf. `http://v2.ocaml.org/releases/4.06/htmlman/libref/Graphics.html`)

```
let triangle = [| (250, 250) ; (400, 250) ; (250, 400) |] ;;

let dessin2 () =
    open_graph "" in
    set_color yellow ;
    fill_circle 80 40 20 ;
    set_color red ;
    fill_poly triangle;
    wait_next_event (Button_down :: []) ;;
```

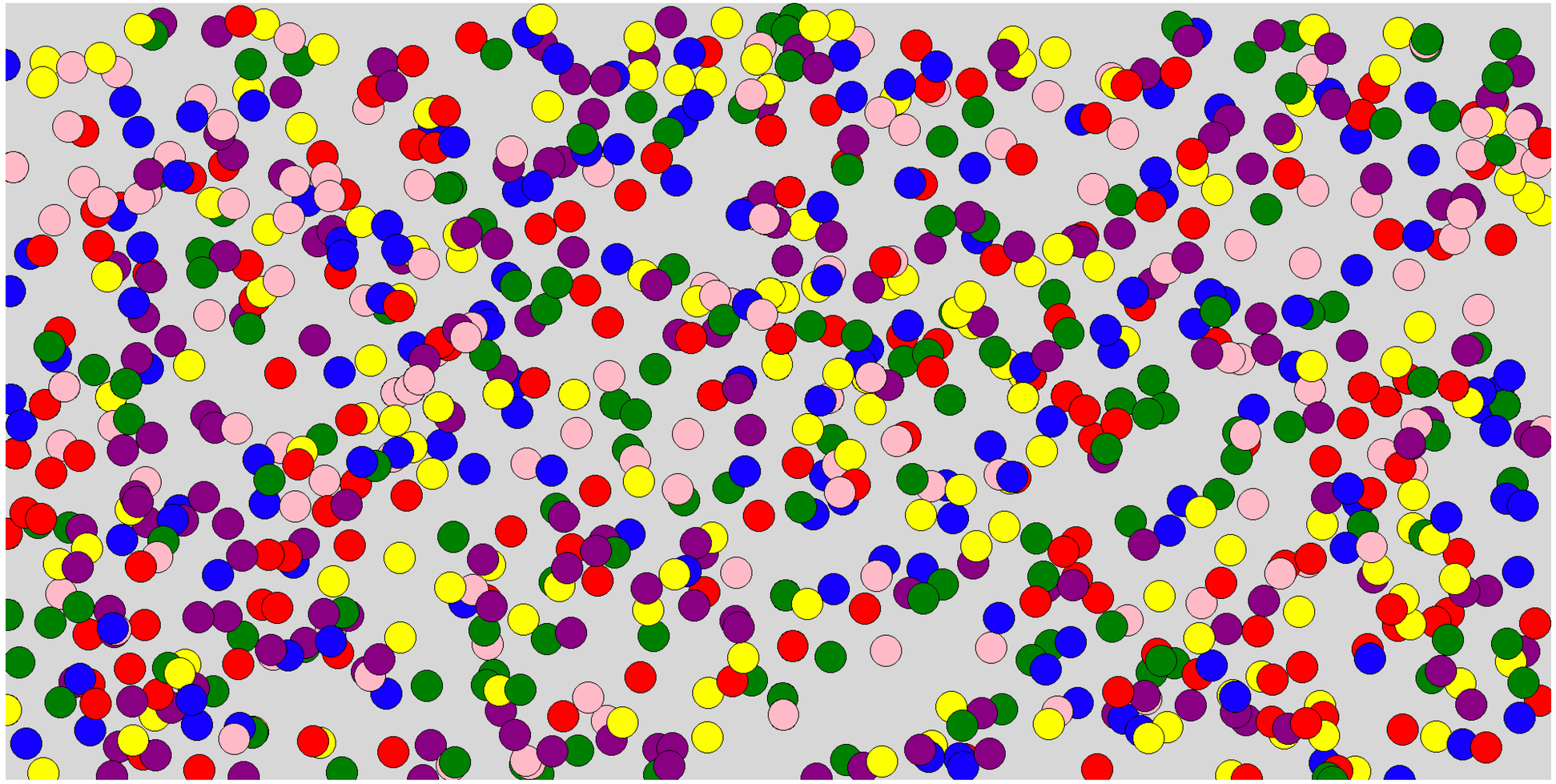


les sources sont en `http://jeanjacqueslevy.net/prog-fp/progs/fp4b.ml`

Graphique

- un paquetage `graphics.cma` simple pour dessins 2D (a besoin d'installer le module `graphics`)

  (cf. `http://v2.ocaml.org/releases/4.06/htmlman/libref/Graphics.html`)



Graphique

- un paquetage `graphics.cma` simple pour dessins 2D (a besoin d'installer le module `graphics`)

  (cf. `http://v2.ocaml.org/releases/4.06/htmlman/libref/Graphics.html`)

```
let dessin3 () =
    open_graph "" ;
    let winx = 1500 and winy = 800 in
    resize_window winx winy ;
    let cols = [|red; yellow; green; blue; purple; pink |] in
    let ncols = Array.length cols in
    set_color lightgrey ;
    fill_rect 0 0 winx winy ;
    let n = 600 in
    for i = 0 to n-1 do
      let c = cols.(Random.int ncols) in
      let a = Random.int (winx-20) in
      let b = Random.int (winy-20) in
      set_color c ;
      fill_rect a b 20 20
    done ;
    wait_next_event (Button_down :: []) ;;
```

- code couleur en RGB

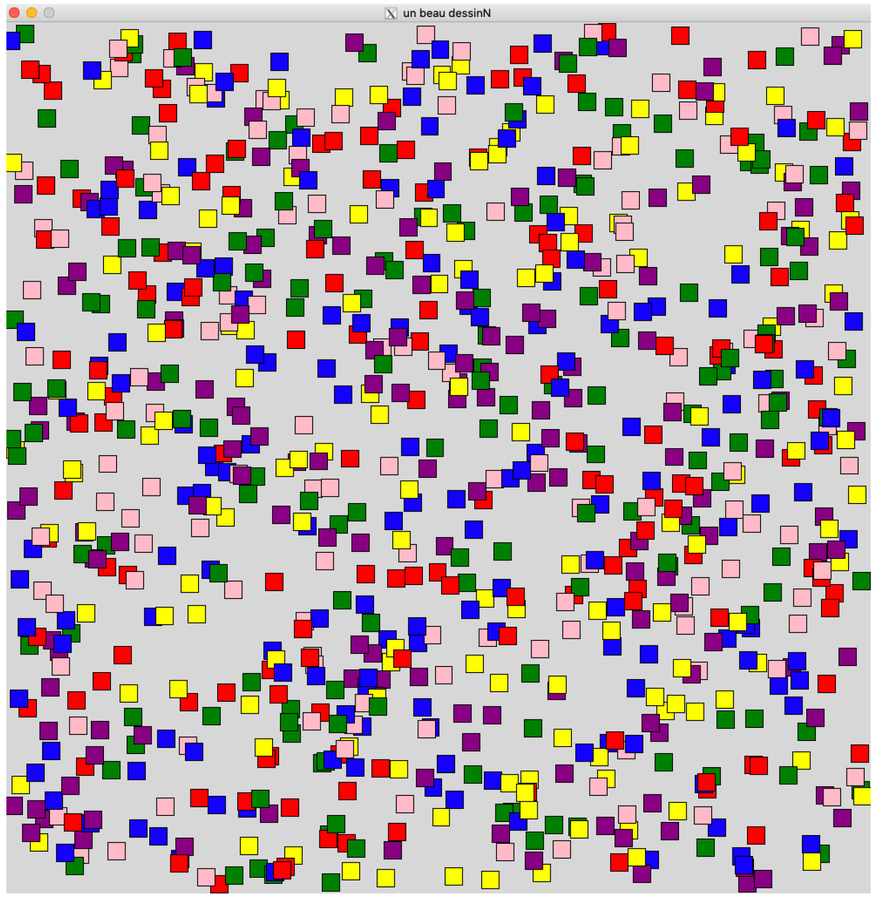
```
let pink = 0xffc0cb ;;
let purple = 0x800080 ;;
let lightgrey= 0xd9d9d9 ;;
```

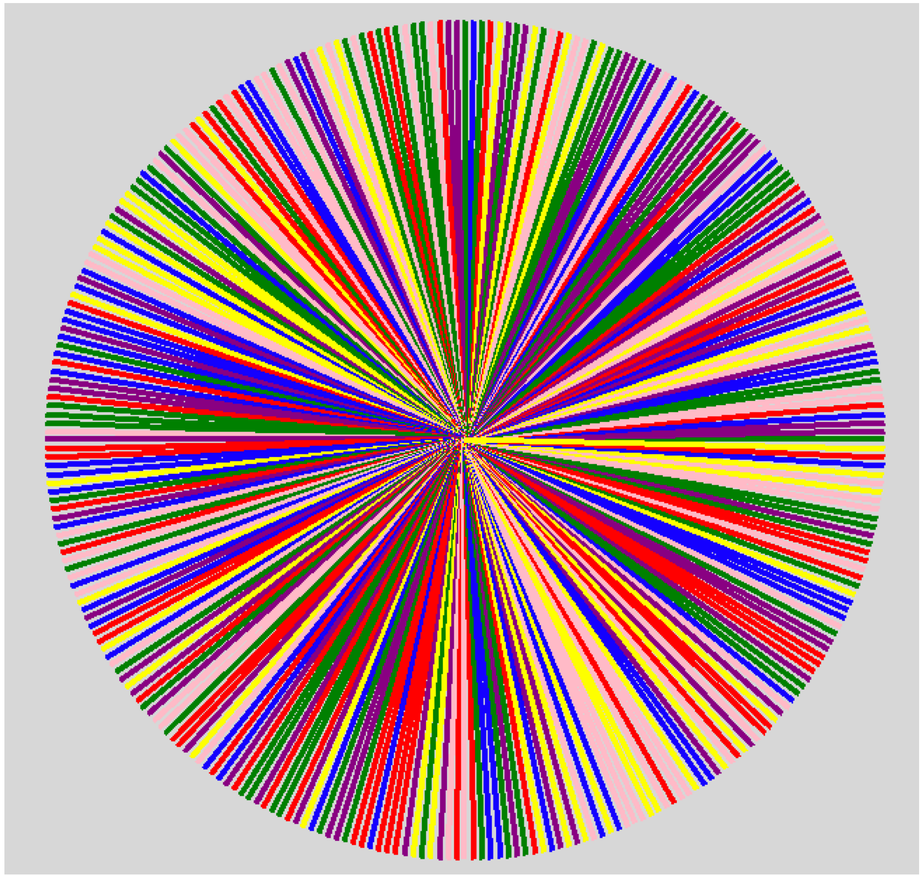
les sources sont en `http://jeanjacqueslevy.net/prog-fp/progs/fp4b.ml`

Graphique

- un paquetage `graphics.cma` simple pour dessins 2D (a besoin d'installer le module `graphics`)

  (cf. `http://v2.ocaml.org/releases/4.06/htmlman/libref/Graphics.html`)



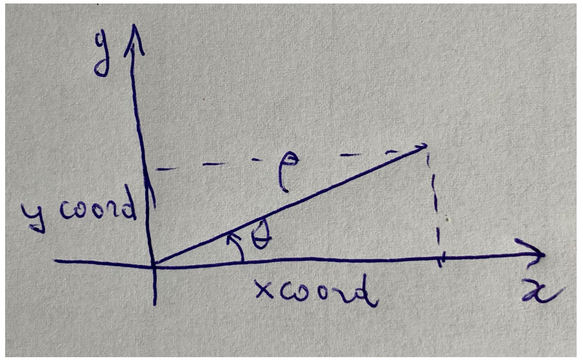


Graphique

- un paquetage `graphics.cma` simple pour dessins 2D (a besoin d'installer le module `graphics`)

(cf. `http://v2.ocaml.org/releases/4.06/htmlman/libref/Graphics.html`)

```
let dessin5 n rho =
    open_graph "" ;
    let winx = 1000 and winy = 800 in
    resize_window winx winy ;
    set_color lightgrey ; fill_rect 0 0 winx winy ;
    let cols = [|red; yellow; green; blue; purple; pink |] in
    let ncols = Array.length cols in
    let pi = Float.pi in
    let theta = 2.0 *. pi /. float_of_int n in
    let centerx = winx / 2 in
    let centery = winy / 2 in
    for i = 0 to n-1 do
      let c = cols.(Random.int ncols) in
      set_color c ;
      let itheta = float_of_int i *. theta in
      moveto centerx centery;
      set_line_width 4 ;
      rlineto (xcoord rho itheta) (ycoord rho itheta);
    done ;
    wait_next_event (Button_down :: []) ;;
```



où

```
let xcoord rho theta = int_of_float ((float_of_int rho) *. (Float.cos theta)) ;;
let ycoord rho theta = int_of_float ((float_of_int rho) *. (Float.sin theta)) ;;
```

Conclusion

VU:

- alias
- références et variables modifiables
- n-uplets et chaînes de caractères
- tris élémentaires
- graphique 2D

TODO list

- récursivité
- listes
- filtrage
- types de données structurées