Noyau fonctionnel

Programme = suite finie (éval. dans l'ordre)

- Expressions → valeurs
- Déclarations → noms

Séparateur (expr) `;;`

Valeurs basiques...
Comparaisons, opérations logiques...
Arithmétique...

Traits impératifs

En vrai, plein de manip. mémoire, bien cachées
(Allocation) / libération ⇝ garbage collector

Traits impératifs pour effets de bord

Effets de bord ⇝ modification mémoire explicite

- Structures mutables
- Références sans accès adresse
- Instructions...

Égalité « physique »

Ici entiers sur 63 bits
Au commencement : `=` ≠ `==`

```
let a = 1::[]
let b = 1::[]
a = b ?
a == b ?
let c = a
a == c ?
```


Objets mutables

```
type toto = { mutable mc1 : int;
              mc2 : int}
```

(Rappel : enregistrement avec accès : •)

Écrire dans le mutable : `<-`

```
let a = { mc1 = 0 ; mc2 = 3}

a.mc1 <- 42
```

Objets mutables

Structures mutables : tableaux

```
let a = Array.make 5 0;;
val a : int array = [|0; 0; 0; 0; 0|]
```

Accès •(indice) écriture : `<-`

```
a.(3);;
- : int = 0

a.(3) <- 42;;
-: unit = ()

a.(3);;
- : int = 42
```


Boucles. . .

`while`/`for`

Sémantique `for` : condition intouchable

```
for i=42 to 666 do ... done
for i=666 downto 42 do ... done
```

Sémantique `while` : modif. possible

```
while a.(3) != a.(0) do ... done
```


Références

Comme enregistrements à un champ mutable `contents`

```
let a = ref 666
```

Affectation par `:=`

```
a := 42;;
- : unit = ()
```

Déréférencement par `!`

```
!a;;
- : int = 42
```


Partager !

Économiser de la mémoire — mais objets plus gros

Gagner du temps sur les comparaisons — mais coût de création

À construction d'une structure :

- Si déjà une identique, alors cette dernière
- Sinon enregistrement de la construction (avec accès rapide)

Partager !

Économiser de la mémoire — mais objets plus gros

Gagner du temps sur les comparaisons — mais coût de création

À construction d'une structure :

- Si déjà une identique, alors cette dernière
- Sinon enregistrement de la construction (avec accès rapide)

$\rightsquigarrow$ besoin structure efficace pour recherche/assoc.