

CAML

6 - Programmation impérative

<http://tsi.tuxfamily.org/OCaml>



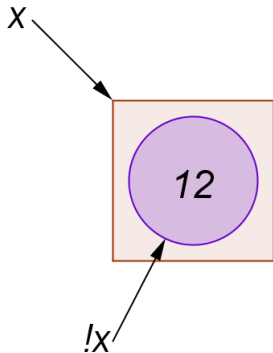
29 mars 2024

Vocabulaire

Déclarer une **référence** permet de nommer un emplacement de la mémoire destiné à y placer des objets qui ont vocation à être modifiés. Il s'agit donc d'une variable **mutable**.

On distingue :

- L'emplacement x .
- du contenu $!x$.



Définir et utiliser **référence** :

```
# let x = ref 3;;  
val x : int ref = {contents = 3}  
# x;;  
- : int ref = {contents = 3}  
# x+1;;  
Characters 1-2:  
  x+1;;  
  ^  
Error: This expression has type int ref  
      but an expression was expected of type int  
# !x+1;;  
- : int = 4
```

Cette expression est de type int ref, mais est utilisée avec le type int.

Modifier une référence :

```
# !x+1;;  
- : int = 4  
# x := !x+1;;  
- : unit = ()  
# x;;  
- : int ref = {contents = 4}  
# !x;;  
- : int = 4
```

Boucle conditionnelles TANT QUE :

```
#let i = ref 0 in
  while !i < 5 do
    print_int !i;
    i := !i + 1;
  done;;
01234- : unit = ()
```

Exercice :

Quel est le type de la fonction calcul ainsi définie? Que renvoie-t-elle?

```
let calcul x y =  
  let a = ref x and b = ref y and tmp = ref 0 in  
    while !b <> 0 do  
      tmp := !a;  
      a := !b;  
      b := !tmp mod !b  
    done;  
  !a  
;;
```

Une exemple de rédaction de la solution : on reconnaît l'algorithme d'Euclide... Démontrons que `calcul x y` renvoie le PGCD de x et y si x et y sont des entiers naturels non simultanément nuls.

Une exemple de rédaction de la solution : on reconnaît l'algorithme d'Euclide... Démontrons que `calcul x y` renvoie le PGCD de x et y si x et y sont des entiers naturels non simultanément nuls.

- Prouvons la terminaison : le variant de boucle est la variable b :
 - ◇ Si $y = 0$ au début, on n'entre pas dans la boucle, ainsi le programme se termine.
 - ◇ Sinon, après le premier passage dans de la boucle, b devient le reste de la division $a \div b$. On prouve ainsi que l'accumulateur b décroît strictement à chaque itération tout en restant dans \mathbb{N} . b fini donc par valoir 0 et la boucle s'arrête.

- Prouvons la correction du programme : l'invariant de boucle pour pourrait être (★) : $PGCD(a, b) = PGCD(x, y)$.
 - ◇ Avant d'entrer dans la boucle, $a = x$ et $b = y$, donc (★) vraie.
 - ◇ Si en entrant dans la boucle, on a (★), notons \tilde{a} et \tilde{b} les valeurs d'entrée de boucle de a et b . A la dernière ligne de la boucle, $a = \tilde{b}$ et il existe $q \in \mathbb{N}$, tel que $b = \tilde{a} - \tilde{b} \times q$. Ainsi $PGCD(a, b) = PGCD(\tilde{b}, \tilde{a} - \tilde{b} \times q) = PGCD(\tilde{b}, \tilde{a}) = PGCD(x, y)$ par hypothèse.
 - ◇ A la fin de la boucle, $b = 0$, or on a toujours $PGCD(x, y) = PGCD(a, b) = PGCD(a, 0) = a$.
L'accumulateur a contient donc bien le $PGCD(a, b)$.

Remarque : si $a = b = 0$, l'algorithme renvoie 0 (On rappelle que le couple $(0, 0)$ n'a pas de plus grand diviseur commun.) et si a ou b est négatif, on peut prouver que les valeurs de b restent décroissantes en valeur absolue, l'algorithme s'arrête donc, cependant, il peut retourner $-PGCD(a, b)$.

Boucle inconditionnelle POUR :

```
for i = 1 to 5 do print_int i; done;;  
#12345- : unit = ()
```



Différences avec Python

Les tableaux

Vocabulaire

Un tableau, aussi appelé "**vecteur**", est une suite finie et modifiable de valeurs d'un même type.

exemple :

3	4	1	0	9
---	---	---	---	---

Les tableaux

Vocabulaire

Un tableau, aussi appelé "**vecteur**", est une suite finie et modifiable de valeurs d'un même type.

exemple :

3	4	1	0	9
---	---	---	---	---

indice : 0 1 2 3 4

- Créer un tableau :

```
# let t = [|1; 3; 2; 7|];;  
val t : int array = [|1; 3; 2; 7|]  
# let s = Array.make 2 "OCaml";;  
val s : string array = [|"OCaml"; "OCaml"|]
```

- Connaitre le nombre d'éléments dans un tableau :

```
# let n = Array.length t;;  
val n : int = 4
```

- Lire la valeur au rang i :

```
# let elm = let t = [|1; 3; 8; 7|] in t.(2);;  
val elm : int = 8
```

ou encore

```
# Array.get [|1; 3; 8; 7|] 2;;  
- : int = 8
```

- Modifier une valeur :

```
# let t = [|1; 3; 2; 7|];;  
val t : int array = [|1; 3; 2; 7|]  
# t.(1) <- 0;;  
- : unit = ()  
# t;;  
- : int array = [|1; 0; 2; 7|]
```

ou encore

```
# Array.set t 3 5;;  
- : unit = ()  
# t;;  
- : int array = [|1; 0; 2; 5|]
```

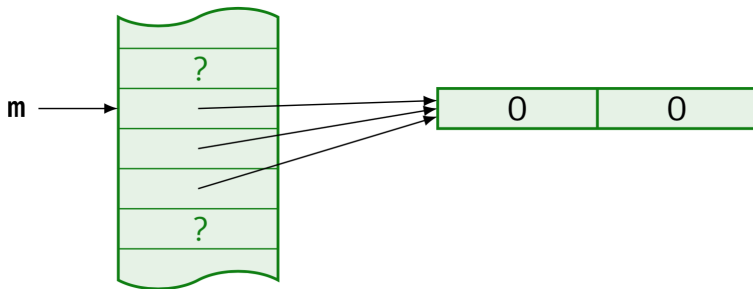

Quelles différences entre les tableaux et les listes ?

Quel type de programmation choisir ?

Pour créer une matrice (tableau à deux dimension), on peut créer un tableau de tableau : une première idée :

```
# let m = Array.make 3 (Array.make 2 0);;
val m : int array array =
      [| [|0; 0|]; [|0; 0|]; [|0; 0|] |]
# m.(0).(0) <- 1;;
- : unit = ()
# m;;
- : int array array =
      [| [|1; 0|]; [|1; 0|]; [|1; 0|] |]
```

Explication :



Deux alternatives :

```
#let m = Array.make 3 [| |];;
val m : '_weak1 array array=[| [| |]; [| |]; [| |] |]

#for i = 0 to 2 do m.(i)<-Array.make 2 0 done;;
- : unit = ()

# m;;
- : int array array =
      [| [|0; 0]|; [|0; 0]|; [|0; 0]| |]

# m.(0).(0) <- 1;;
- : unit = ()

# m;;
- : int array array =
      [| [|1; 0]|; [|0; 0]|; [|0; 0]| |]
```

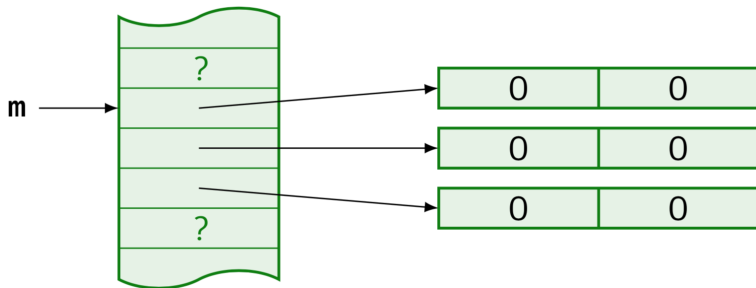
ou en utilisant la fonction `matrix` qui fait le travail pour nous :

```
# let m = Array.make_matrix 3 2 0;;
val m : int array array =
      [| [|0; 0|]; [|0; 0|]; [|0; 0|] |]

# m.(0).(0) <- 1;;
- : unit = ()

# m;;
- : int array array =
      [| [|1; 0|]; [|0; 0|]; [|0; 0|] |]
```

Cette fois-ci :



Sur un exemple : on cherche à déterminer la i -ième plus grande valeur d'un tableau d'**entiers naturels** et renvoie -1 si ce tableau contient moins de i valeurs.

Sur un exemple : on cherche à déterminer la i -ième plus grande valeur d'un tableau d'**entiers naturels** et renvoie -1 si ce tableau contient moins de i valeurs.

L'algorithme proposé ici consiste à répéter i fois :

- Récupérer la plus grande valeur et son indice dans le tableau.
- Remplacer cette valeur par -1

A la fin, on renvoie la plus grande valeur trouvée au dernier tour.

Voici une proposition de code :

```
let plusgrand t i =
  for i = 1 to i do
    let maxi = ref (-1) and imax = ref (-1) in
      for j = 0 to Array.length t - 1 do
        if t.(j) > !maxi then begin
          maxi := t.(j);
          imax := j
        end;
      done;
    t.(!imax) <- -1
  done;
  !maxi
;;
```

Pourquoi cette solution ne fonctionne pas ?

Pourquoi cette solution ne fonctionne pas ?

- ▶ Problème de portée de variable.

Pourquoi cette solution ne fonctionne pas ?

► Problème de portée de variable.

Que se passe-t-il si on exécute :

```
let t = [|3;1;9;7;4|];;  
plusgrand t 2;;  
t;;
```

Pourquoi cette solution ne fonctionne pas ?

- ▶ Problème de portée de variable.

Que se passe-t-il si on exécute :

```
let t = [|3;1;9;7;4|];;  
plusgrand t 2;;  
t;;
```

- ▶ Problème d'effet de bord (side effect).

OCaml distingue :

- Le caractère 'a'
- La chaîne de caractères "a"

```
# let x = 'a';;  
val x : char = 'a'  
# let x = "a";;  
val x : string = "a"
```



Différences avec Python

Opérations sur les chaînes : on utilise le module `String` d'où la syntaxe (proche de Python) :

- Concaténation : ^

```
# "bon" ^ "jour";;  
- : string = "bonjour"
```

- Créer une chaîne de longueur donnée :

```
# String.make 3 'a';;  
- : string = "aaa"
```

(Le second argument est un caractère et non une chaîne).

- Connaître la longueur d'une chaîne :

```
# String.length "OCaml";;  
- : int = 5
```

- Extraire une sous-chaîne :

```
# let t = "OCaml est un langage fonctionnel"  
  in String.sub t 13 7;;  
- : string = "langage"
```

- Connaître le caractère d'indice i :

```
# let txt = "OCaml" in txt.[1];;  
- : char = 'C'
```


- Obtenir le code ASCII d'un caractère et inversement :

```
#int_of_char `A`;;  
- : int = 65  
#char_of_int 70;;  
- : char = `F`
```

American Standard Code for Information Interchange
Code américain normalisé pour l'échange d'information

x1 x10	0	1	2	3	4	5	6	7	8	9
3			espace	!	"	#	\$	%	&	'
4	()	*	+	,	-	.	/	0	1
5	2	3	4	5	6	7	8	9	:	;
6	<	=	>	?	@	A	B	C	D	E
7	F	G	H	I	J	K	L	M	N	O
8	P	Q	R	S	T	U	V	W	X	Y
9	Z	[\]	^	_	`	a	b	c
10	d	e	f	g	h	i	j	k	l	m
11	n	o	p	q	r	s	t	u	v	w
12	x	y	z	{		}	~	□	€	

Conversions :

- `int_of_string "100"` renvoie `100`
- `float_of_string "100"` renvoie `100.0`
- `string_of_int 100` renvoie `"100"`
- `string_of_float 100.0` renvoie `"100.0"`

Plus généralement, `type1_of_type2` convertit un élément de `type2` en un élément de `type1`.

- Modifier le caractère d'indice i : Une chaîne est une donnée **non mutable**. Depuis la version 4.02 d'OCaml, on ne peut plus modifier un élément d'une chaîne (comme en Python).
- Néanmoins, un autre type : `Bytes` représentant une suite **mutable** d'octets de longueur fixée permet de contourner le problème si besoin. Les octets sont donnés sous forme de caractères en correspondance avec leurs codes ASCII.

```
# let txt = Bytes.of_string "CAML";;  
val txt : bytes = Bytes.of_string "CAML"  
# Bytes.get txt 1;;  
- : char = 'A'  
# Bytes.set txt 3 'E';;  
- : unit = ()  
# Bytes.to_string txt;;  
- : string = "CAME"
```

Ce type n'est pas à connaître, si vous aviez à l'utiliser dans un sujet, les différentes fonctions liées à ce type seraient données en énoncé