Introduction à OCaml

Exploration des structures de données usuelles

MPSI - Prytanée National Militaire

Pascal Delahaye

28 mars 2018

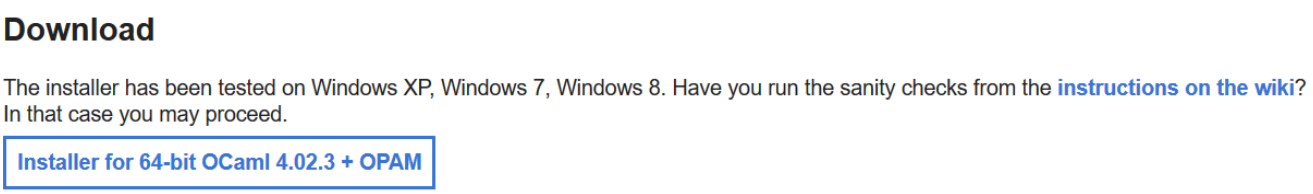
Le langage informatique utilisé dans ce cours est OCaml (Objective Categorical Abstract Machine - Meta-Language). Ce premier cours donne une présentation rapide de ce langage.

1 Installation sous windows

L'installation sous windows se fait en deux étapes :

1. Installation du compilateur et des bibliothèques OCaml :
   (a) Aller à l'adresse suivante : https ://caml.inria.fr/download.fr.html

   (b) Dans la page qui s'ouvre sélectionner :



2. Installation de l'interface graphique wincaml : http ://jean.mouric.pagesperso-orange.fr/

3. Paramétrage de wincaml : sélectionner l'option "OCaml" dans le menu "CamlTop"

2 Un langage fortement typé

Tous les objets manipulés par OCaml ont un *type*.
Les types les plus fréquemment rencontrés sont les suivants :

Type	code	exemple 1	exemple 2
Nombre entier	int	2	-2
Nombre flottants (réels)	float	3.	sqrt 3.
Caractère	char	'h'	
chaîne de caractère	string	"vive caml"	"Erreur d'argument"
Liste	type1 list	[1 ; 2]	[2.1 ; sqrt 3. ; atan 1.]
Tableau	type1 array	[|1 ; 2|]	[|2.1 ; sqrt 3. ; atan 1.|]
Type inconnu 1	'a	[ ]	[| |]
Type inconnu 2	'b	[ ], [ ]	[| |], [| |]
Booléen	bool	true	(1=2)
Fonction d'une variable	type1 − > type2	int_of_float	min
Action OCaml	unit	print_int	v.(2)< −1

Remarque 1.

1. Ces types vous apparaitront peut-être contraignants, mais ils aident le programmeur en :
   (a) l'obligeant à identifier systématiquement la nature des objets qu'il manipule.
   (b) lui donnant des informations sur les erreurs de programmation éventuelles.
2. Le type `unit` correspond à une modification du contenu des variables ou une action de type "affichage à l'écran".
3. Il existe encore bien d'autres types que l'on rencontrera progressivement au fil des cours.

2.1 Les types de nombres

Exercice : 1

Familiarisation avec les types `int` et `float`
Taper les instructions suivantes et commenter la réponse de Caml dans le tableau suivant :

Instructions	Réponse	Commentaires
let a = 2 ; ;		
let a = 3. ; ;		
let a = 1 + 2 ; ;		
let a = 1 + 2.1 ; ;		
let a = 1.2 + 2.1 ; ;		
let a = 1.2 +. 2.1 ; ;		

Remarque 2.

1. Une série de commandes OCaml se termine toujours par `;;`
2. La création (on dit aussi "la déclaration") d'une variable (qui s'exprime sous la forme > `nom = ...` sous Python) commence ici toujours par `let nom = ....`
3. Les noms de variables ne doivent jamais commencer par une majuscule.
4. On prendra l'habitude de mettre des espaces avant et après chaque opérateur.

A retenir ! !

1. On ne peut faire d'opérations qu'entre éléments de même type. Ainsi :

2. Les opérateurs associés aux entiers sont différents des opérateurs associés aux réels ! !

Opérateurs entre entiers :	+	-	/	*	(pas d'opérateur "puissance")
Opérateurs entre flottants :	+.	-.	/.	*.	**

3. Les fonctions associées aux entiers sont différentes des fonctions associées aux réels ! !

Fonctions sur des entiers :	abs	mod	pred	succ	min	max	float_of_int	Random.int
Fonctions sur des réels :	abs_float	exp	log	sqrt	min	max	int_of_float	Random.float
	sin	sinh	asin					

Remarque 3. Contrairement à Python, les fonctions mathématiques usuelles sont directement accessibles sous OCaml, sans avoir besoin de les importer depuis une bibliothèque.

Exercice : 2

Tester (durant 5 mn) les différents opérateurs et fonctions précédentes.

Remarque 4.

1. Les fonctions `min` et `max` s'appliquent aussi à des couples ...
2. ⚠ Division euclidienne sur les entiers : $\begin{cases} \texttt{a mod b} \text{ donne le reste} \\ \texttt{a/b} \text{ donne le quotient} \end{cases}$ de la division euclidienne de a par b.

Représentation machine des nombres sous OCaml

1. Les nombres entiers sont codés en binaire sur 63 bits, le premier bit correspondant au signe.
   Cela signifie qu'il est possible de manipuler des entiers compris entre -2^{62} et $2^{62}-1$.

   Comparer les valeurs de 2^{62} et de -2^{62} (utiliser `int_of_float`).
   Que constatez-vous ? Comment interprétez-vous ce résultat ?

   La manipulation informatique de grands nombres entiers peut donc donner des résultats abberrants !

2. Les nombres réels (appelés *flottants*) sont également codés sur 64 bits par :
   - 1 bit pour le signe,
   - 52 bits pour la mantisse
   - 11 bits pour l'exposant

Que ce soit pour les entiers ou les nombres flottants, les valeurs limites seront donc très rarement atteintes.

Remarque 5.

1. Caml n'est pas un programme de calcul formel.
   - Les nombres non entiers sont donc approchés par des nombres flottants.
   - Le nombre π pourra être défini de façon approchée par : `let pi = 4.*.atan(1.);;`

2. Vous avez constaté qu'il n'existe malheureusement pas d'opérateur *puissance* sur les entiers.

2.2 Listes indexées et dynamiques

Exercice : 3

Familiarisation avec les types `liste` et `array`

Taper les instructions suivantes et commenter la réponse de Caml dans le tableau suivant :

Instructions	Réponse	Commentaires
`let l1 = [1;2;4;8];;`		
`List.length l1;;`		
`let l2 = [1,2,4,8];;`		
`List.length l2;;`		
`let v1 = [|1;2;4;8|];;`		
`Array.length v1;;`		
`let v2 = [|1,2,4,8|];;`		
`Array.length v2;;`		
`l1.(1);;`		
`v1.(1);;`		
`v1.(1) <- 0;;`		
`v1;;`		
`let l3 = [1;2.];;`		
`let v3 = [|1;2.|];;`		

A retenir pour l'instant ! !

1. Les éléments d'un tableau ou d'un vecteur sont tous du même type.
2. Les éléments d'un tableau de longueur n sont numérotés de 0 à $n-1$.
3. Les longueurs d'une liste et d'un tableau sont données par les fonctions `Array.length` et `List.length`
4. On NE peut PAS récupérer facilement une composante quelconque d'une liste.
5. On NE peut PAS modifier les composantes d'une liste.
6. On PEUT récupérer facilement une composante quelconque d'un tableau.
7. On PEUT modifier les composantes d'un tableau.

Remarque 6. Nous verrons plus tard que :

- le type TABLEAU sera utilisé en programmation itérative
- le type LISTE sera surtout utilisé en programmation récursive.

Exercice : 4

Compléments sur le type `liste`

Taper les instructions suivantes et commenter la réponse de Caml dans le tableau suivant :

Instructions	Réponse	Commentaires

Instructions	Réponse	Commentaires
`let l1 = [1;2;4;8];;`		
`let t::q = l1;;`		
`t;; q;;`		
`List.hd l1;;`		
`List.tl l1;;`		
`let l2 = 0::l1;;`		
`let l3 = l1@l2;;`		

A retenir pour l'instant !!

1. On ne peut pas obtenir directement la valeur d'une composante quelconque d'une liste.
2. On obtient la valeur de la tête et de la queue d'une liste `l` :
   - Soit à l'aide de la commande `let t::q = l;;`
   - Soit à l'aide des fonctions `t = List.hd l` et `q = List.tl l`

   Pour faciliter la compréhension, on conviendra de noter `t` la valeur de tête et `q` la queue d'une liste.
3. On peut ajouter une valeur `a` en tête d'une liste `l` par la commande `let l = a::l;;`
4. On peut concaténer une liste `l1` à une liste `l2` par la commande `let l = l1@l2;;`

`Exemple 1.` Déterminer deux algorithmes très simples permettant de :

1. extraire la p^{eme} composante d'une liste
2. modifier la p^{eme} composante d'une liste

`Exemple 2.` Comme en Python, on pourra utiliser la structure de Tableau pour définir une matrice.

Créer en langage OCaml la variable T définie par : $T = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \end{pmatrix}$.

2.3 Les types de "mots"

Exercice : 5

Familiarisation avec les types `char` et `string`

Taper les instructions suivantes et commenter la réponse de Caml dans le tableau suivant :

Instructions	Réponse	Commentaires
`let debut = 'l';;`		
`let fin = "'addition svp !";;`		
`let phrase = debut ^  fin;;`		
`let phrase = "l" ^  fin;;`		
`String.length phrase;;`		

A retenir pour l'instant ! !

1. Les caractères et les chaînes de caractères ont des types différents. (`char` et `string`)
2. On peut concaténer deux chaînes de caractères entre elles par la commande `let chaîne = ch1 ^ ch2;;`
3. La fonction `String.length` permet de déterminer la longueur d'une chaîne de caractères.

Remarque 7. **Complément :**
Les chaînes de caractères se comportent comme des tableaux dont les éléments sont des caractères.

1. On obtient le n^{eme} caractère d'une chaîne "nom" par la commande : `nom.[n-1];;`
2. On peut modifier le n^{eme} caractère d'une chaîne "nom" par la commande : `nom.[n-1] <- 'A';;`

Appliquer ces manipulations sur des exemples de votre choix !

2.4 Les tuples

A partir des types de base, on peut générer de nouveaux types à la façon des produits cartésiens de types existants.

Exercice : 6
Familiarisation avec les types `tuple`
Taper les instructions suivantes et commenter la réponse de Caml dans le tableau suivant :

Instructions	Réponse	Commentaires
let couple1 = (1,2) ; ;		
let couple2 = (3,1) ; ;		
couple1 = couple2 ; ;		
couple1 > couple2 ; ;		
min couple1 couple2 ; ;		
let uplet1 = ("toto", 2, false) ; ;		
let uplet1 = "toto", 2, false ; ;		

A retenir pour l'instant ! !

1. Il est possible de créer des tuples de longueur quelconque
2. Les parenthèses ne sont pas indispensables mais sont très utiles pour garantir la lisibilité des commandes.
3. Les composantes des tuples peuvent être de types différents
4. Il est possible d'effectuer des tests de comparaison sur des tuples d'entiers ou de flottants.
5. Les tuples seront surtout utilisés dans les fonctions pour renvoyer plusieurs valeurs.

2.5 Les booléens

Les booléens sont utilisés dès lors que la réalisation d'une commande est conditionnelle à la vérification d'une condition (de nature booléenne).

Exercice : 7
Familiarisation avec le type `booléen`
Taper les instructions suivantes et commenter la réponse de Caml dans le tableau suivant :

Instructions	Réponse	Commentaires

Instructions	Réponse	Commentaires
`let bool1 = (2>=3);;`		
`let bool2 = ("toti"<>"toto");;`		
`bool1 && bool2;;`		
`bool1 || bool2;;`		
`let bool3 = not (3 < 2);;`		

A retenir pour l'instant ! !

1. le booléen "a et b" s'écrit `a && b`
2. le booléen "a ou b" s'écrit `a || b`
3. le booléen "contraire de a" s'écrit `not a`
4. Pour comparer des $\left\{ \begin{array}{l} \text{entiers} \\ \text{flottants} \end{array} \right.$ on utilise les comparateurs standards : $<$, $>$, $=$, $>=$, $<=$ et $<>$
5. Pour comparer des chaînes de caractères on utilise les comparateurs standards : $=$ et $<>$. Il en existe d'autres basés sur l'ordre lexicographique.

2.6 Modification du type d'une variable

Il est possible de transformer le type d'un élément par la fonction `type1_of_type2`.

Exercice : 8

Transformer le type d'éléments de votre choix.

Transformation du type	Code	résultat
Entier en réel	`float_of_int`	
Réel en entier	`int_of_float`	
Réel en chaîne	`string_of_float`	
Entier en chaîne	`string_of_int`	
chaîne en entier	`int_of_string`	
chaîne en réel	`float_of_string`	
Booléen en chaîne	`string_of_bool`	
Code ASCI d'un caractère	`int_of_char`	
Caractère associé à un code ASCI	`char_of_int`	

A retenir pour l'instant ! !

1. On peut changer le type d'un élément de `type2` en `type1` avec `type1_of_type2`.
2. On ne peut pas changer n'importe quoi en n'importe quoi.
3. Mais attention au sens des commandes `int_of_char` et `char_of_int` ! !

`Exemple 3.`

1. Déterminer le code ASCII du caractère `@`
2. Déterminer le caractère dont le code ASCII est 35

Les caractères d'imprimerie ainsi que la tabulation et le "retour chariot" sont tous codés en code ASCII sur 8 bits. Cela donne en tout 256 caractères. Vérifiez cela à l'aide de `char_of_int`.

Remarque 8. Nous verrons que OCaml permet à l'utilisateur qui le désire, de créer ses propres types.

3 Déclaration des constantes

Les constantes sont des variables dont la valeur reste constante au cours du projet.
Il y a deux types de constantes :

1. Les constantes globales
2. Les constantes locales

Exercice : 9

Familiarisation avec les deux types de constantes
Taper les instructions suivantes et commenter la réponse de Caml dans le tableau suivant :

Instructions	Réponse	Commentaires
`let a = 25;;`		
`let b = 2 in 3*b;;`		
`a;;`		
`b;;`		

A retenir ! !

1. Les constantes globales sont définies par l'instruction `let nom = valeur ;;`
2. Les constantes locales sont définies par l'instruction `let nom = valeur in ... ;;`
3. Vous rencontrerez parfois le message d'erreur : `The value identifier b is unbound.`
   Il signifie que la variable `b` est inconnue de OCaml (et donc que sa déclaration a échoué).

Exercice : 10

Déclaration de plusieures constantes locales :
Taper les instructions suivantes et commenter la réponse de Caml dans le tableau suivant :

Instructions	Réponse	Commentaires
`let a = 2 and b = 3 in a + b;;`		
`let a = 2 and b = 3*a in a + b;;`		
`let a = 2 in let b = 3*a;;`		
`let b = let a = 2 in 3*a;;`		
`let a = 2 in let b = 3*a in a + b;;`		

A retenir ! !

1. Une variable globale b dépendant d'une autre variable A se déclare de la façon suivante :
   `let b = let a = ... in ...`
2. On peut créer des constantes locales à des constantes locales.
   Pour cela on utilise la syntaxe : `let nom1 = valeur1 in let nom2 = valeur2 in ..procedure.. ;;`
   Ici, la variable `nom1` est locale à la constante `nom2` qui elle-même est locale à la procédure.
3. On utilise la commande `and` lorsqu'on souhaite déclarer plusieures constantes locales de même niveau.

Remarque 9. Si l'on souhaite ajouter des commentaires à un programme, on utilisera la syntaxe `(* .... *)`

Instructions	Réponse	Commentaires
(* je déclare une constante globale a *) let a = 2 ; ;		

Remarque 10. Les commentaires sont indispensables pour garantir la bonne lisibilité d'un programme.