Les commandes essentielles en OCaml

Déclarations et instructions

commentaires	(* commenter le code*)
définition d'une valeur	`let v=2`, `let f x=2*x+4`
définition récursive	`let rec f =...`
définition locale	`let a=3 in`
définitions parallèles	`let a=... and b=...`
définitions successives	`let a=... in let b=...`
variable modifiable	`let v= ref ...`
valeur d'une référence	`!v`
modification d'une référence	`v:=...`
fonction sans argument	`let f ()= ...`
fonction à un argument	`let f x=...`
fonction à plusieurs arguments	`let f x1 x2=...`
expression conditionnelle	`if condition then exp1 else exp2`

choix multiple	`match valeur with`
	`| motif-1 -> exp1`
	`|motif-2 -> exp2`
	`|_ -> exp3`

ne rien faire	`()`
calculs en séquence	`begin ...end`
boucle croissante	`for i=début to fin do ...done`
boucle décroissante	`for i=début downto fin do ...done`
boucle conditionnelle	`while condition do ...done`
déclencher une erreur	`failwith "message"`

Expressions booléennes

vrai, faux	`true, false`
ou	`||`
et	`&&`
non	`not`

Expressions entières

opérations arithmétiques	`+ - * /`
modulo	`mod`
valeur absolue	`abs`
entier précédent, suivant	`pred succ`
min ,max	`min a b, max a b`
entier aléatoire entre 0 et n-1	`Random.int(n)`

Expressions réelles

opérations arithmétiques	`+. -. *. /.`
puissance	`**`
min ,max	`min a b, max a b`
fonctions mathématiques	`abs_float exp log sqrt sin cos tan` `sinh cosh tanh asin acos atan`
réel - > entier	`int_of_float`
entier - > réel	`float`
réel - > chaine	`string_of_float`
chaine -> réel	`float_of_string`
réel aléatoire entre 0 et a	`Random.float(a)`

Listes

liste composée de x,y, z,...	`[x;y;z;..]`
liste vide	`[ ]`
ajouter a en tête de l	`a : : l`
tête et queue de l	`List.hd l`, `List.tl l`
longueur de l	`List.length l` (*complexité linéaire*)
concaténation de l1 et l2	`l1@l2` (*complexité linéaire*)
image miroir de l	`List.rev l` (*complexité linéaire*)

Tableaux

tableau composé de x,y,z,...	`[|x;y;z;...|]`
tableau vide	`[| |]`
kième élément de tab	`tab.(k)`
modification du kième élément de tab	`tab.(k)<-x`
longueur de tab	`Array.length tab`
création d'un tableau contenant n fois x	`Array.make n x`
création d'une matrice	`Array.make_matrix n p x`
extraction d'un sous tableau (indices de i à i+long-1)	`Array.sub tab i long`

Chaines de caractères

caractère	`'e'`
chaine de caractères	`"bonjour"`
kième caractère	`chaine.[k]`
modification	`chaine.[k] <- 'e'`
longueur d'une chaine	`String.length`
extraction	`String.sub chaine deb long`
concaténation	`chaine1^chaine2`