

TD1 : Premiers Pas en OBJECTIVE CAML

Objectifs Maîtriser la règle de portée d'OBJECTIVE CAML, déterminer le type d'une expression, et savoir écrire des fonctions (récursives) simples.

1 Règle de portée

Exercice 1 Vérifier le respect des règles de portée sur les expressions suivantes. Si l'expression est typable, donner son type et sa valeur.

- | | |
|--|--|
| <p>1. <code>let x = 1 in
let y = 2 in
let x = 3 in
x + y</code></p> <p>2. <code>let x =
let y = 1 in
y + 1
in x * 2</code></p> <p>3. <code>let x =
let y = 1 in
y + 1
in x + y</code></p> <p>4. <code>let x =
let y = 1 in y + 1
in
let y = 2 in x + y</code></p> <p>5. <code>let x =
let y = 1 in
x + y
in x * 2</code></p> | <p>6. <code>let x = 1 in
let x = true in x</code></p> <p>7. <code>let x =
if true then
let tmp = 5.0 in
tmp /. 2.0
else 0.0
in x</code></p> <p>8. <code>let x =
if true then
let tmp = 5.0 in
0.0
else tmp /. 2.0
in x</code></p> <p>9. <code>let x = 1;;
let x = 2 in x;;
let y = x in y;;</code></p> <p>10. <code>let f =
let x = 1
let y = 2 in
x
in f</code></p> |
|--|--|

2 Fonctions et fonctions n-aires

Exercice 2 Donner le type des fonctions suivantes ou expliquer pourquoi elles ne sont pas typables.

- | | |
|--|--|
| <p>1. <code>let f x = x + 1</code></p> <p>2. <code>let f x = x = 1 x = 1.0</code></p> <p>3. <code>let f x y = x + y</code></p> <p>4. <code>let f x y z w = x + y + z * w</code></p> | <p>5. <code>let f x =
let y = 2 * x in
if x = 0 then max_int
else x + y</code></p> |
|--|--|

```

6. let abs x =
    if x < 0 then -x else x
7. let incr = fun x -> x + 1
8. let incr_or_decr b =
    if b then (fun x -> x + 1)
    else (fun x -> x - 1)
9. let g y =
    let f x = x * y in
    f 0 + f 1
10. let f x =
    let y = x * x in
    print_int y

```

Exercice 3 Étant donnée la fonction suivante :

```

let affiche opt x =
  if opt then print_int x
  else ()

```

Déterminer les applications de fonctions suivantes qui sont valides.

1. `affiche true 42`
2. `affiche (true, 42)`
3. `(affiche true) 42`

Exercice 4 Écrire le corps des fonctions suivantes.

1. La fonction `xor x y` doit implémenter le ou exclusif.
`val xor : bool -> bool -> bool`
2. La fonction `max x y` doit donner le maximum de deux entiers.
`val max : int -> int -> int`
3. On vous donne la fonction `String.length : string -> int`; implémenter une fonction imprimant la longueur d'une chaîne :
`val affiche_longueur : string -> unit`
4. On vous donne la fonction `String.sub : string -> int -> int -> string`.
`String.sub s pos len` renvoie la sous chaîne de `s` de longueur `len` commençant en `pos`. Implémenter une fonction `affiche_tronque n s` qui affiche les `n` premiers caractères de `s` :
`val affiche_tronque : int -> string -> unit`

3 Fonctions récursives

Exercice 5 Écrire le corps des fonctions suivantes.

1. La fonction `fibonacci`: `val fact : int -> int` qui calcule la valeur, pour un entier donné, de la suite de fibonacci définie par :

$$\begin{aligned}
 fib(0) &= 0 \\
 fib(1) &= 1 \\
 fib(n) &= fib(n-1) + fib(n-2)
 \end{aligned}$$
2. Le plus grand dénominateur commun de deux entiers: `val pgcd : int -> int -> int`
3. On vous donne les fonctions `Char.chr : int -> char` et `print_char : char -> unit`. Écrire la fonction `affiche_ascii n m` qui affiche les caractères ASCII entre `n` et `m` :
`val affiche_ascii : int -> int -> unit`

Exercice 6 Écrire trois fonctions différentes de multiplication de deux entiers.