Programmation fonctionnelle : feuille d'exercices 1

María-Virginia Aponte

28 février 2011

Exercice 1

Expressions

Chacune des phrases Ocaml dans cet exercice est une *expression*. *Rappel* : Cela signifie en particulier que l'on peut appliquer des opérateurs ou des fonctions sur ces expressions ou les passer en paramètre.

On utilise ici les fonctions prédéfinies suivantes :
- `succ` : prend en argument un entier n en renvoie en résultat l'entier n+1 ;
- `Char.lowercase` : prend un caractère c et renvoie ce caractère convertit en majuscule ;
- `Char.code` : prend en argument un caractère et renvoie son code ASCII ;
- `Char.chr` : prend en argument un entier correspondant à un code ASCII et renvoie le caractère associé.

Question 1

1. Devinez les valeurs et types des expressions, puis comparez votre réponse avec celle donnée par Ocaml.
2. Classifiez ces expressions. S'agît-il d'une expression arithmétique, de comparaison, un appel de fonction, une expression conditionnelle ?

```
1  #  3 + 7;;
2  - : int = 10
3
4  #  3. +. 7.5;;
5  - : float = 10.5
6
7  # "bonjour" = "Bonjour";;
8  - : bool = false
9
10 # "bonjour" < "salut";;
11 - : bool = true
12
13 # "bonjour" < "salut" || "bonjour" = "Bonjour" ;;
14 - : bool = true
15
16 # succ 4;;
17 - : int = 5
18
19 # (succ 4) * 5;;
20 - : int = 25
21
22 # succ 4 * 5;;
23 - : int = 25
```

```
24
25 #  succ (4 * 5);;
26 - : int = 21
27
28 # succ(succ 5) + 3;;
29 - : int = 10
30
31 # Char.lowercase 'A';;
32 - : char = 'a'
33
34 #  Char.code(Char.lowercase 'A');;
35 - : int = 97
36
37 # Char.chr(Char.code(Char.lowercase 'A') +1);;
38 - : char = 'b'
39
40 # if 4=5 then succ 3 else succ 4;;
41 - : int = 5
42
43 #  (if 4=5 then succ 3 else succ 4) + 2;;
44 - : int = 7
45
46 #  succ (if 4=5 then succ 3 else succ 4);;
47 - : int = 5
48
49 # (if 4=5 then succ 3 else succ 4) = 6;;
50 - : bool = false
51
52 # ((if 4 >= 4 then succ 3 else succ 4) > 6) = ('a' > 'b') ;;
53 - : bool = true
54
55 # let a = 3 in (a>3) = (a<3);;
56 - : bool = true
57
58 # (let a = 3 in (a+3))*4;;
59 - : int = 24
60
61 # succ (let a = 3 in (a+3));;
62 - : int = 7
```

Question 2

Ces expressions sont mal typées. En vous aidant de la réponse donnée par Ocaml, expliquez pourquoi.

```
1  # 2 + 3.0;;
2  This expression has type float but is here used with type int
3
4  # succ 3.0;;
5  This expression has type float but is here used with type int
6
7  # 1 < 1.5;;
8  This expression has type float but is here used with type int
9
10 # if 4=4 then 1 else true;;
```

```
11 This expression has type bool but is here used with type int
12
13 # succ 3 4;;
14 Error: This function is applied to too many arguments;
15 maybe you forgot a ';'
```

Exercice 2

Déclarations, expressions.

Classifiez les phrases suivantes en déclarations ou expressions. (*Rappel* : les déclarations locales sont des expresssions). Expliquez les réponses données par Ocaml.

```
1  # let x = 2 in x+3;;
2  - : int = 5
3
4  # x+3;;
5  Unbound value x
6
7  # (let x=2 in x+3) + (let z=4 in z);;
8  - : int = 9
9
10 #  (let x=2 in x+3) + (let z=4 in z+x);;
11 Unbound value x
12
13 # let x= let y=3 in y;;
14 val x : int = 3
15
16 # x;;
17 - : int = 3
18
19 # let cube x = x*x*x;;
20 val cube : int -> int = <fun>
21
22 # cube 2;;
23 - : int = 8
24
25 # let double_cube x = 2*(cube x);;
26 val double_cube : int -> int = <fun>
27
28 # double_cube 2;;
29 - : int = 16
```

Exercice 3

Typage

La fonction `String.length` calcule la longueur d'une chaîne de caractères. Quelles sont les réponses données par Ocaml ?

```
# String.length "Bonjour" - 1;;
```

```
# (String.length "Bonjour") - 1;;

# String.length ("Bonjour" - 1);;

# let moyenne x y = (x+.y) /. 2.0;;

# moyenne 4.0 6.0;;

# moyenne (4.0 6.0);;

# let test x = if x>0 then 1 else "1";;

# let f x = not x;;

#f not true;;

# f (not true);;
```

Exercice 4

Bien comprendre les déclarations locales.

Question 1

Dans le code suivant il y a des phrases de syntaxe très proches mais pour lesquelles les messages Ocaml sont différents. Expliquez pourquoi.

```
# (if 3>7 then 1 else 10) + 5;;
- : int = 15

# let a = (if 3>7 then 1 else 10) + 5;;
val a : int = 15

# let b = 1 in let c = b;;
Error: Syntax error

# let b = 1 in let c = b in b=c;;
- : bool = true

# b;;
Error: Unbound value b

# let b = let c = 3 in c+5;;
val b : int = 8

# b;;
- : int = 8
```

Question 2

Que fait la fonction `f` ? La fonction `g` est-elle locale ou globale ? Que fait-elle ? Testez la fonction `f` sur deux exemples différents.

```
# let f x =
```

```
  let g y = Char.code y
  in Char.chr (succ (g x));;
val f : char -> char = <fun>
```

Exercice 5

Premières fonctions
Considérez les déclarations de fonctions suivantes :

```
# let minuscule c = ('a' <= c) && (c <= 'z');;
val minuscule : char -> bool = <fun>

# let majuscule c =  ('A' <= c) && (c <= 'Z');;
val majuscule : char -> bool = <fun>

# let lettre c = minuscule c || majuscule c;;
val lettre : char -> bool = <fun>
```

1. Que font ces fonctions ? Testez les dans la boucle interactive d'Ocaml en donnant à chacune successivement le caractères 'b', 'B', '*' et '4' en argument.
2. Donnez une définition de la fonction `lettre` où les fonctions `majuscule` et `minuscule` sont déclarées localement.
3. Donnez une définition de la fonction `lettre` sans faire appel à d'autres fonctions.
4. Donnez une définition d'une fonction `chiffre` qui teste si un caractère est un chiffre. Exemple : `chiffre '5'` répond true et `chiffre 'a'` répond false.

Exercice 6

Écrire des fonctions.
L'appel `(a mod b)` calcule le reste de la division entière de a par b. Par exemple :

```
# 7 mod 3;;
- : int = 1
# 6 mod 3;;
- : int = 0
# 2 mod 3;;
- : int = 2
```

1. Á l'aide de cette fonction, écrivez une fonction `divisible` qui prend deux entiers a et b, et teste si a est divisible par b. Testez votre fonction sur plusieurs exemples pour vous assurer qu'elle fonctionne correctement.
2. Utilisez la fonction `divisible` pour écrire une fonction `pair` qui prend en argument un entier a et teste s'il est pair.

Exercice 7

Écrire des fonctions
L'appel `float_of_int n` convertit un entier n dans son équivalent de type `float`. Par exemple :

```
# float_of_int;;
- : int -> float = <fun>
# float_of_int 2;;
- : float = 2.
# 3.5 +. 2;;
This expression has type int but is here used with type float
# 3.5 +. (float_of_int 2);;
- : float = 5.5
```

1. Expliquez les messages donnés par Ocaml pour le code précèdent.
2. Utilisez la fonction `float_of_int` pour écrire une fonction `inverse` qui prend en argument un entier x et calcule son inverse $1/x$. Votre fonction doit calculer un `float`. Exemple : `inverse 2` doit répondre `- : float = 0.5`
3. Testez votre fonction avec l'argument 0. Expliquez le message affiché par Ocaml.

Exercice 8

L'appel `int_of_char c` renvoie le code ASCII d'un caractère c (l'équivalnt de Char.code). Par exemple :

```
# int_of_char 'a';;
- : int = 97
# int_of_char '0';;
- : int = 48
```

1. Expliquez les messages donnés par Ocaml pour le code précèdent.
2. Utilisez la fonction `int_of_char` pour écrire une fonction `car_vers_chiffre` qui prend en argument un caractère c qui est un chiffre et calcule l'entier qu'il représente. Si le caractère n'est pas un chiffre, la fonction renvoie son caractère unicode. Exemple : `car_vers_chiffre '2'` doit répondre `- : int = 2`, et `car_vers_chiffre 'a'` doit répondre `- : int = 97`.
3. Testez votre fonction sur plusieurs sortes de caractères.