Resumen del estilo imperativo de Ocaml

Jorge Alberto Chávez sarmiento

Resumen

Los apuntes que se presentan son unas notas breves de clase, no pretenden ser una guia para aprender ocaml, son mas bien una pequeña guia para recordar los comandos más básicos de la parte imperativa de este lenguaje.

Código 1: Algunos ejemplos

```
1 # let b=3;;
2 # let g x = b*x;;
3 # g 4;;
4 -: int = 12
5 # let b = 4;;
6 # g 4;;
7 -: int =12 ( \neq 16)
```

1 Referencias

Código 2: Uso de referencias

```
1 let b = ref 3;;
2 val b = int ref ={contents = 3}
```

como 3 es de tipo int, el tipo de la casilla b, es el tipo de las referencias que contienen int.
Se nota int ref
El contenido puede cambiar pero el tipo no. “ref” es un constructor de tipo (como list, -> , arbol_bin)
para hablar del contenido de b: !b

Código 3: Ejemplos de referencias

```
1 !3;;
2 -: int=3;;
3 # 3+!b;;
4 -:int=6
5 # let g x = !b*x;;
6 # g 4;;
7 -: int = 12
```

Modificar el contenido de b:

Código 4: Modificando contenido de referencias

```
1 b:= 4;;
2 significa: el nuevo contenido de b va a ser 4
3 -:unit =( )
4 # !b;;
5 -: int = 4;;
6 # g 4;;
7 -: int = 16
8 # b:='hola';;
9 error: tipo string en vez de int.
```

2 Funciones:

Definir una función para agregar 1 al contenido del parámetro.

Código 5: Función Incremento

```
# let incr r = r := !r+1;;
val incr : int ref -> unit = < fun >
# incr b;;
-: unit = ( )
# !b;;
-: int = 5
```

Las funciones para incrementar y decrementar ya vienen definidas en caml por incr y decr, respectivamente.

3 Valores compartidos

Código 6: Valores compartidos

```
# let r1=ref 1;;
val r1: int ref = {contents = 1}
# let r2 = r1;;
(es la misma casilla)
# r2:= 2;;
-:unit = ( )
# r1;;
int ref = {contents = 2}
# r1:= 4;;
# !r2;;
-: int = 4
```

4 Impresión a la pantalla

No existe función general:

Código 7: Impresión en pantalla

```
print_int: int -> unit
print_float: float -> unit
print_string: string -> unit
print_char: char -> unit
print_newline: unit -> unit ( para ejecutar esta funcion print_newline ( ) )
```

`\n`

es un carácter especial para saltar una línea.

Código 8: Ejemplo de impresión en pantalla

```
# print_string "hola!\n";;
hola!
- : unit = ( )
```

5 Secuencias

En ocaml la sucesion se nota “;” (punto y coma) y se usa para dar varias ordenes seguidas.
Ejemplo:

Código 9: Ejemplo de secuencia

```
# print_int 3; print newline ();;
3
-: unit = ( )
```

los **;** se pueden encadenar: exp1; exp2; exp3; ... ;expn
Ejercicio: Escribir una función que recibe una cadena y que imprime esta cadena entre dos líneas blancas.

Código 10: Resolución del ejercicio

```
#let imprime c = print_newline ();
print_string c;
print_newline ();
print_newline ();;
```

6 Ciclos

Sirve para repetir una secuencia de cálculos muchas veces.

6.1 Ciclo while

En ocaml: While condicion do exp done
Ejemplo: (Factorial)

Código 11: Factorial de un número

```
#let rec fact = 0 -> 1|n -> n*fact(n-1);;
n!=1x2x3x4x ... x n
#let fact n = let i = ref 1 and p = ref 1 in
while !i <= n do
p := !p*!i;
i := !i +1; (este punto y coma es opcional) (se puede escribir tambien como incr i)
done;
!p;;
```

Ejercicio: Escribir la función M.C.D. Con el estilo imperativo.

Código 12: Máximo común divisor

```
mcd (a,0)=a
mcd(a,b)=mcd (b, a mod b)
En ocaml:
#let mcd a b = let x= ref a and let y= ref b in
while !y != 0 do
let temp = !x in
x:=!y;
y:=temp mod !y;
done;
!x;;
```

6.2 Ciclo for

Cuando un índice involucra un índice, es más práctico usar el ciclo for.
for indice = val1 to val2 do exp done
Por ejemplo para el factorial:

Código 13: Factorial de un número

```
let fact n= let p= ref 1 in
for i= 1 to n do
p:= !p*i
done;
!p;;
```

7 Arreglos (vectores o tablas)

| 0 | 1 | 2 | 3 | 4 | ... | n-1 |
Ejemplo:

Código 14: Modificando contenido de referencias

```
#let v=[|6;5;8|];;
val v: int array = [|6;5;8|]
Otra forma:
#let v2 = array.make 6 4;;
val v2 = int array = [|4;4;4;4;4;4|]
```

- la función array.make a b crea un arreglo de tamaño a con todos los valores iguales a b.
- la función array.length permite conocer la longitud de un arreglo.

```
#array.lenght v;;
-: int 3
```

Para consultar las casillas:
Globalmente:

```
#v;;
-: int array = [|6;5;61|]
```

Individualmente:

```
#v.(0);;
-:int = 6
#v.(2);;
-: int =8;;
#v.(-2);;
exepcion: invalid argument ‘‘index out of bounds’’
```

Ejemplo:
Calcular el elemento más pequeño de un arreglo

Código 15: Elemento más pequeño de un arreglo

```
#let rec min_vect2 v i j = if i = j then t.(i)
 else min (t.(i)) (min:vect2 v (i+1)j);;
#let min vect v= let n= array.length v in
 if n = 0 then failwith "arreglo vacio"
 else min vect2 v 0 (n-1);;
```

Modificación de casillas:
Se usa

```
v.(i) <- exp
```

para modificar la casilla i de v
Ejemplo:

Código 16: Modificando casillas

```
#v.(1) <- 10;;
#v;;
-: int array =[|6;10;8|]
```

8 Cadenas de caracteres como vectores

Una cadena (string) puede ser ”vista“ como un vector de caracteres, las operaciones de creación, acceso, modificación son parecidas a las de los vectores pero como string es diferente que char array, la sintaxis es distinta: los paréntesis se cambian a corchetes cuadrados.
Ejemplo:

Código 17: Cadenas como vectores

```
#let s= "Hola␣Roberto"
val s= string= "Hola␣Roberto"
# s.[11] <- 'a';;
-: unit = ( )
# s;;
-:string = "Hola␣Roberta"
String.make 10 'B'
-: string = "BBBBBBBBBB"
# String.length s
-:int = 12
```

9 Matrices

Sea M una matriz nxp, n lineas y p columnas. Podemos ver M como un vector de n lineas cada linea siendo como un vector de p elementos. Su tipo es int array array No hay sintaxis particular a las matrices, sin embargo existe una función de creación:

`Array.make_matrix`

Ejemplo:

Código 18: Ejemplos de matrices

```
# let m1=[|[|1;2;3|];
 [|4;5;6|]|];;
# let m2=Array.make_matrix 2 3 0;;
matriz ( 000
000)
-: int array array= [|[|0;0;0|];[|0;0;0|]|]
# m1.(0).(2);;
-: int = 3
# Array.length m1;;
-: int = 2
# Array.length m1.(0)
-: int = 3
(numero de filas)
( numero de elementos de una fila , o sea numero de columnas)
# m.(1).(0) <- 7;;
# m1;;
-: int array array = [|[|1;2;3|];[|7;5;6|]|];;
```

10 Ciclos y vectores

Ejemplo: escribir una función para imprimir las cadenas de un vector entre comas.

Código 19: Ejemplo de vectores

```
Imprimir_comas [| "uno"; "dos" "tres"|];;
uno, dos, tres
-: unit = ( )
# let imprimir_comas v =
let n=Array.lenght v in
for i=0 to n-2 do
print_string v.(i);
print_string ","
done;
print_string v.(n-1);
print_newline ( );;
```

Ejercicio: Escribir una función que calcule el producto escalar de 2 vectores. **Ejercicio:** Escribir una función que imprime una matriz a la pantalla en forma rectangular. **Ejercicio:** Escribir una función que calcula el producto de dos matrices

Código 20: Producto de matrices

```
let prodcto_matrices a b=
let la=Array.length a in
let ca=Array.length a.(0) in
let lb=Array.length b in
let cb=Array.lenght b.(0) in
if ca=lb then
let c=Array.make_matrix la lc 0. in
for i=0 to la-1 do
for j=0 to cb-1 do
for k=0 to ca-1 do
c.(i).(j)+a.(i).(k)*.b.(k).(j);
done;
done;
done;
;;
```