Linguagens e Ambientes de Programação (Aula Teórica 2)

LEI - Licenciatura em Engenharia Informática

João Costa Seco (joao.seco@fct.unl.pt)

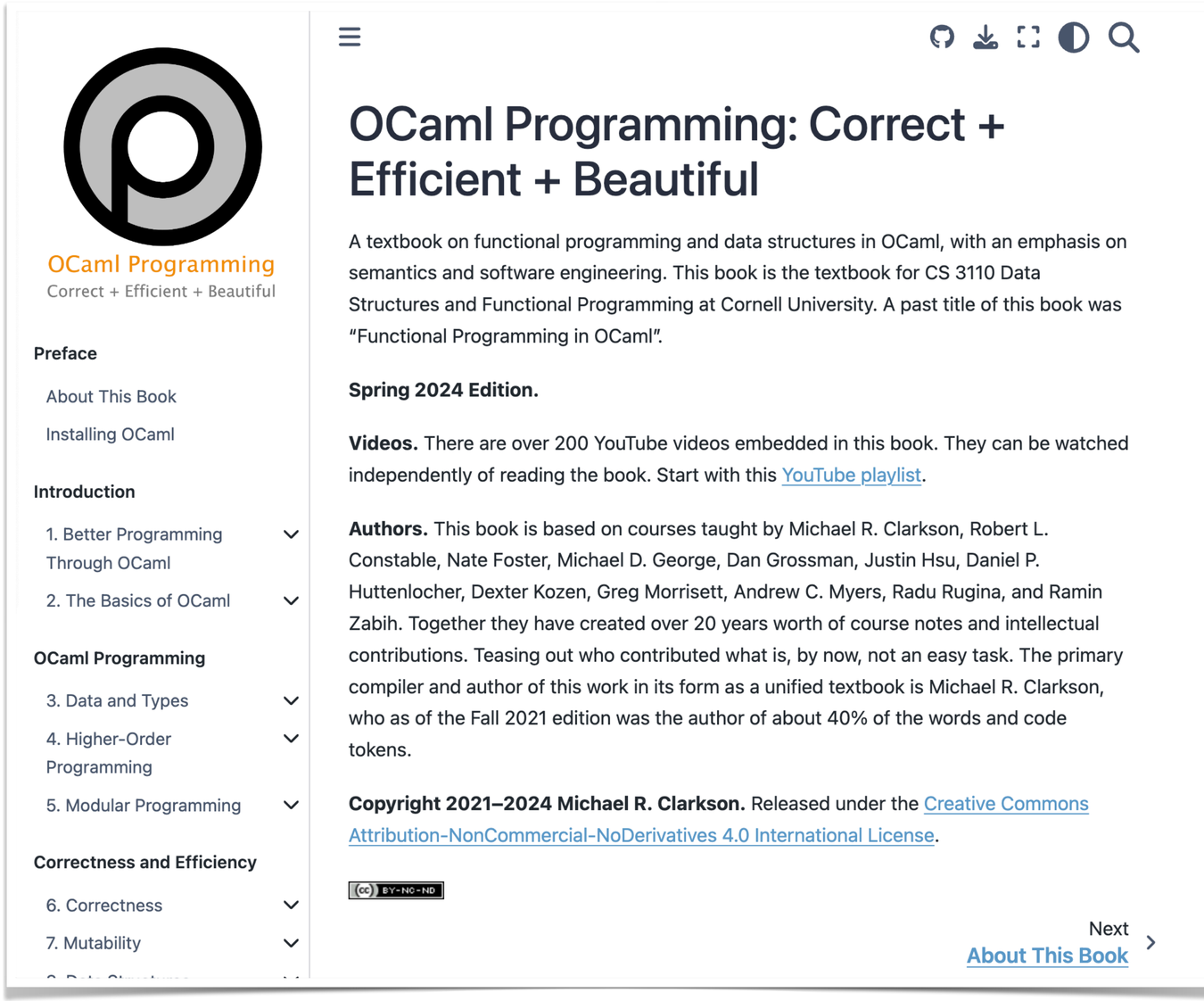
NOVA SCHOOL OF
SCIENCE & TECHNOLOGY

Agenda para hoje

- Introdução à programação funcional.
- A linguagem OCaml.
- Expressões, variáveis e tipos.
- Funções biblioteca. Input/Output básico.

Livro de texto

- OCaml Programming:
  Correct + Efficient + Beautiful
- Cornell University
- Michael R. Clarkson et al.
- Online resources
  (book, exercises, videos)



Principais Características da Programação Funcional

- É uma programação declarativa (especificar “o que é", em vez de “como é”)
- As construções típicas são:
  - Todas as construções são expressões
  - Funções como valores da linguagem
  - Composicionalidade
  - Valores imutáveis
  - Sistema de tipos forte
- Construções usadas noutras linguagens.

“A language that doesn’t affect the way you think about programming is not worth knowing.”

—Alan J. Perlis (1922-1990), first recipient of the Turing Award

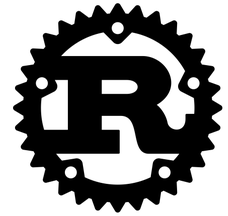
OCaml Programming

Correct + Efficient + Beautiful

Gradiente do paradigma funcional em linguagens

- Linguagens funcionais puras - Haskell
- Linguagens funcionais com características imperativas (estado) - OCaml
- Linguagens imperativas com mecanismos funcionais - Rust
- Linguagens imperativas (procedimentais ou object oriented) - Java<17






Linguagens funcionais (logo game)

ML (OCaml) is logic in programming

$$\frac{A \longrightarrow B \qquad A}{B}$$

ML (OCaml) is logic in programming

$$\frac{f : A \longrightarrow B \quad x : A}{f\ x : B}$$

Example in Java

```
B f1(A a) {...}

C f2(B b) {...}

D f3(C c) {...}

D d = f3(f2(f1(new A())))
```

$$f1 : A \longrightarrow B$$
$$f2 : B \longrightarrow C$$
$$f3 : C \longrightarrow D$$

$$\dfrac{\dfrac{\dfrac{A \quad A \longrightarrow B}{B} \quad B \longrightarrow C}{C} \quad C \longrightarrow D}{D}$$

OCaml: uma linguagem de expressões

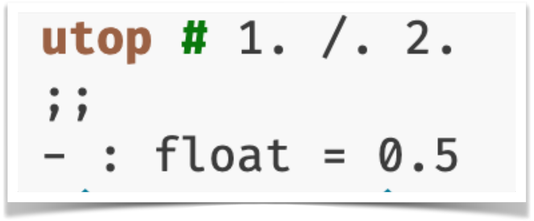
- É um idioma da linguagem ML
- É uma linguagem de expressões que:
  - Ou denotam um valor,
  - Ou terminal com uma exceção
  - Ou não terminam…
- É fortemente tipificada com inferência de tipos

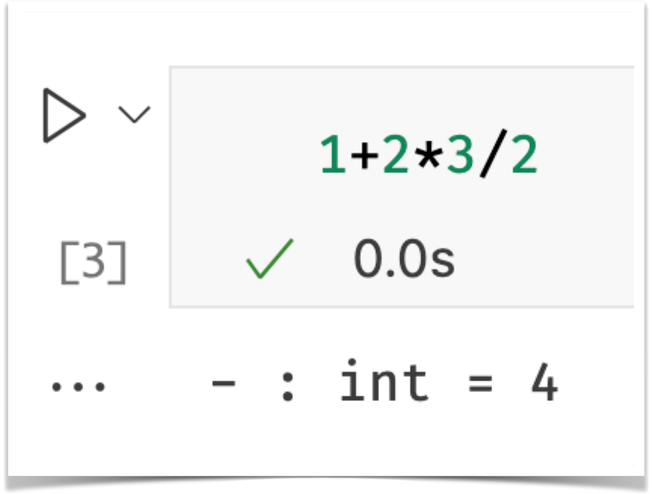
facebook Microsoft docker

Jane Street Bloomberg ahrefs

Ambientes de programação

- Interpretador: ocaml / utop
- Compilador: ocamlc + make/dune
- Visual Studio Code + OCaml plugin
- Jupyter Notebook + OCaml kernel





Tipos básicos, literais, operadores e funções

int	1 2 3 4	+ - * /	
float	1. 2. 3.5 4e10	+. -. *.	float_of_int
bool	true false	&& ||	
char	'a' 'b'		char_of_int ,
strings	"hello" ""	^	string_of_int
unit	()		

```
3.14 * 2.
```

[4] ⊗ ✧ 0.0s

```
File "[4]", line 1, characters 0-4:
1 | 3.14 * 2.
    ^^^^
Error: This expression has type float but an expression was expected of type
         int
```

Tipos básicos, literais, operadores e funções

int	1 2 3 4	+ - * /	
float	1. 2. 3.5 4e10	+. -. *.	float_of_int
bool	true false	&& ||	
char	'a' 'b'		char_of_int ,
strings	"hello" ""	^	string_of_int
unit	()		

```
3.14 *. (float_of_int 2)
```

[5] ✓ 0.0s

```
- : float = 6.28
```

Tipos básicos, literais, operadores e funções

int	1 2 3 4	+ - * /	
float	1. 2. 3.5 4e10	+. -. *.	float_of_int
bool	true false	&& ||	
char	'a' 'b'		char_of_int ,
strings	"hello" ""	^	string_of_int
unit	()		

```
int_of_string "not an int"
```

[6] ⊗ ✧ 0.9s

```
Exception: Failure "int_of_string".
Raised by primitive operation at unknown location
Called from Stdlib__fun.protect in file "fun.ml", line 33, characters 8-15
Re-raised at Stdlib__fun.protect in file "fun.ml", line 38, characters 6-52
Called from Toploop.load_lambda in file "toplevel/toploop.ml", line 212, characters 4-150
```

Tipos básicos, literais, operadores e funções

int	1 2 3 4	+ - * /	
float	1. 2. 3.5 4e10	+. -. *.	float_of_int
bool	true false	&& ||	
char	'a' 'b'		char_of_int ,
strings	"hello" ""	^	string_of_int
unit	()		

```
"abc".[0]
```

[7] ✓ 0.0s

```
- : char = 'a'
```

Igual e Igual

=	<>	structural
==	!=	physical (references, arrays, etc.)

Assertions

- assertions are an effective way of testing functionality.

```
[8]  assert (int_of_string "42" = 43)
     ⊗ ✧ 0.0s

...  Exception: Assert_failure ("[8]", 1, 0).
     Called from Stdlib__fun.protect in file "fun.ml", line 33, characters 8-15
     Re-raised at Stdlib__fun.protect in file "fun.ml", line 38, characters 6-52
     Called from Toploop.load_lambda in file "toplevel/toploop.ml", line 212, characters 4-150
```

Expressões condicionais

- as expressões condicionais são expressões onde os dois ramos têm o mesmo tipo

```
[9]  4 + (if 'a' = 'b' then 1 else 2)
     ✓ 0.0s
...  - : int = 6
```

- o ramo “else” é obrigatório para todos os tipos excepto **unit**

```
[10] if "hello" = "world" then 1
     ⊗ 0.0s
...  File "[10]", line 1, characters 26-27:
     1 | if "hello" = "world" then 1
                                 ^

     Error: This expression has type int but an expression was expected of type
              unit
            because it is in the result of a conditional with no else branch
```

Declaração de variáveis (matemáticas)

- declarações top-level

```
let x = 42
```
[12] ✓ 0.0s

```
val x : int = 42
```

- declarações locais

```
let x = 42 in (string_of_int x)^": the ultimate question of life, the universe, and everything"
```
[13] ✓ 0.0s

```
- : string =
"42: the ultimate question of life, the universe, and everything"
```

```
let x = 1 in (let x = 2 in x + 1) + x
```
[14] ✓ 0.0s

```
- : int = 4
```

```
let x = 1 in let y = 2 in x + y
```
[15] ✓ 0.0s

```
- : int = 3
```

Declaração de variáveis (âmbito)

- A declaração de um nome (x) é limitada ao corpo da declaração (e2)
- o nome (x) não é visível na expressão que define o valor (e1)

```
let x=e1 in e2
```

```
let y=let y=y+1 in y+1 in let y=y+2 in y+2
```

- As declarações obedecem ao princípio da irrelevância dos nomes onde os nomes escolhidos não devem ser relevantes para a avaliação de uma expressão.

Declaração e chamada (aplicação) de funções

- A declaração de um nome (f) é limitada ao corpo da declaração (e2)
- o nome (f) não é visível na expressão que define o valor (e1)

```
let f x = x+1 in f 1
```

Declaração recursivas (âmbito)

- A declaração de um nome (x) é visível no corpo da declaração (e2) e na expressão de definição do nome (e1).

```
let rec x = e1 in e2
```

```
let rec fact x = if x = 0 then 1 else x * fact(x-1)
```

Declaração recursivas (âmbito)

- A declaração de um nome (x) é visível no corpo da declaração (e2) e na expressão de definição do nome (e1).

```
let rec x = e1 in e2
```

```
(* pre: x ⩾ 1
   returns true if x is even false otherwise *)
let rec even x = if x = 0 then true else if x = 1 then false else odd(x-1)

(* pre: x ⩾ 1
   returns true if x is odd false otherwise *)
and odd x = if x = 0 then false else if x = 1 then true else even(x-1)
```

Declarações mutuamente recursivas em C

- Tem que se declarar a função sem a definir.

```
bool odd(int x);

bool even(int x) {
  if( x == 0 ) {
    return false;
  } else if( x == 1 ) {
    return false;
  } else {
    return odd(x-1);
  }
}
```

```
bool odd(int x) {
  if( x == 0 ) {
    return false;
  } else if( x == 1 ) {
    return true;
  } else {
    return even(x-1);
  }
}
```