

Linguagens e Ambientes de Programação

(Aula Teórica 2)

LEI - Licenciatura em Engenharia Informática

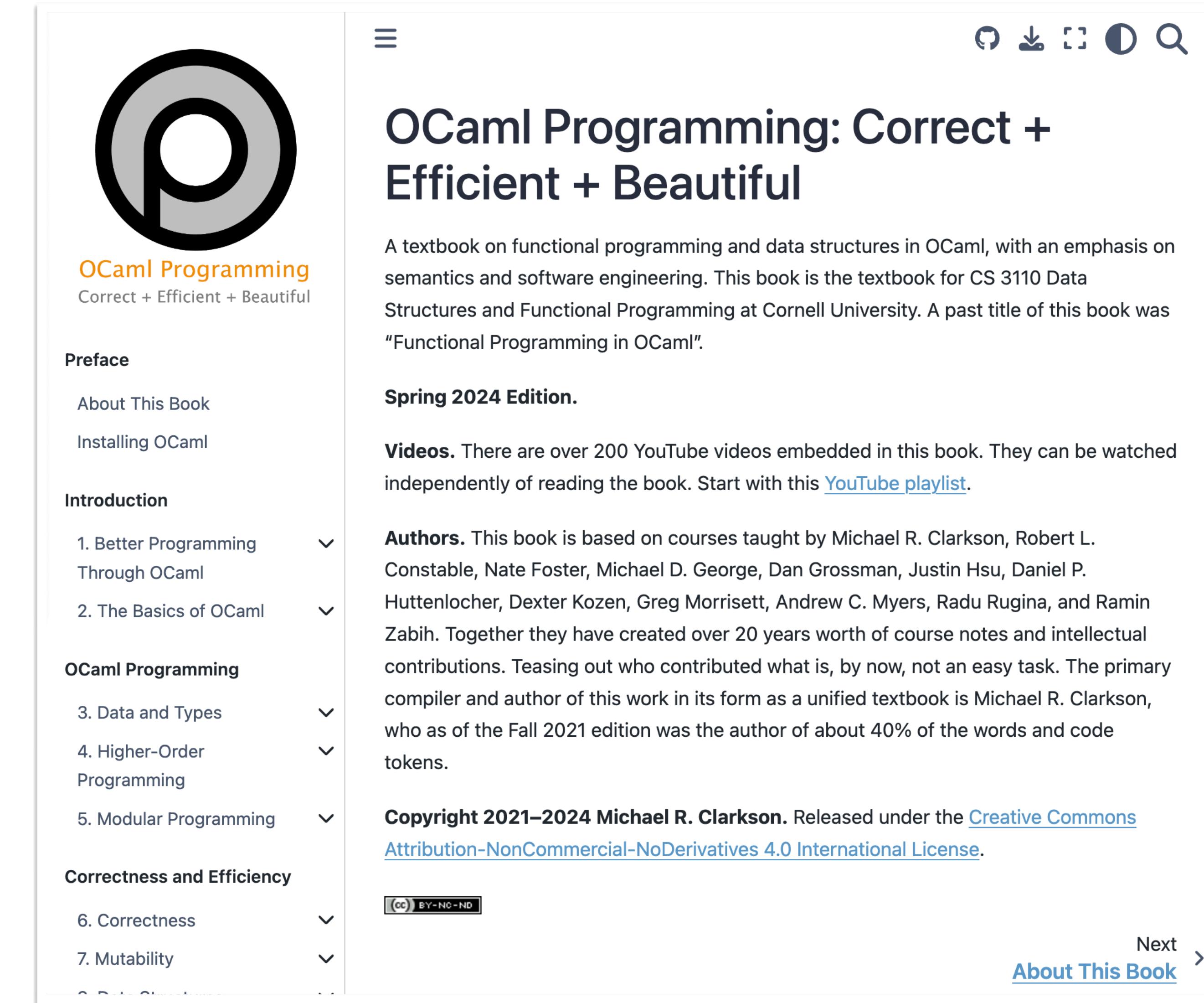
João Costa Seco (joao.seco@fct.unl.pt)

Agenda para hoje

- Introdução à programação funcional.
- A linguagem OCaml.
- Expressões, variáveis e tipos.

Livro de texto

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



The screenshot shows the front page of the OCaml Programming book. At the top right are GitHub, download, and search icons. The title "OCaml Programming: Correct + Efficient + Beautiful" is centered. Below it is a large circular logo with a stylized 'p'. The subtitle "A textbook on functional programming and data structures in OCaml, with an emphasis on semantics and software engineering." is followed by a note about it being the textbook for CS 3110 Data Structures and Functional Programming at Cornell University. It also mentions a past title, "Functional Programming in OCaml". A "Spring 2024 Edition" notice is present. The main content area is organized into sections: Preface, Introduction, OCaml Programming, and Correctness and Efficiency, each with numbered sub-sections. A Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License logo is at the bottom. Navigation links "Next >" and "About This Book" are at the bottom right.

OCaml Programming: Correct + Efficient + Beautiful

A textbook on functional programming and data structures in OCaml, with an emphasis on semantics and software engineering. This book is the textbook for CS 3110 Data Structures and Functional Programming at Cornell University. A past title of this book was "Functional Programming in OCaml".

Preface

About This Book

Installing OCaml

Introduction

1. Better Programming Through OCaml

2. The Basics of OCaml

OCaml Programming

3. Data and Types

4. Higher-Order Programming

5. Modular Programming

Correctness and Efficiency

6. Correctness

7. Mutability

CC BY-NC-ND

Next >

About This Book

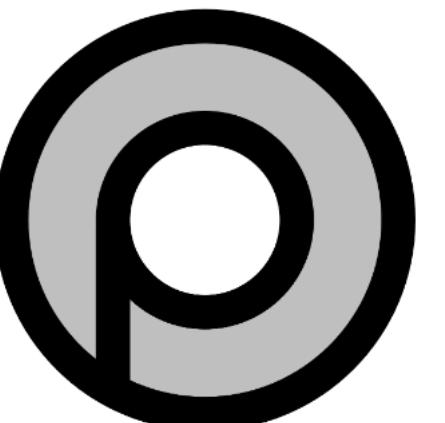
Principais Características da Programação Funcional

- 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.

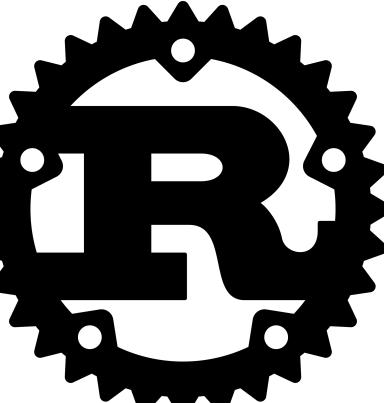


**“A language that doesn’t affect the way you think about
programming is not worth knowing.”**

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



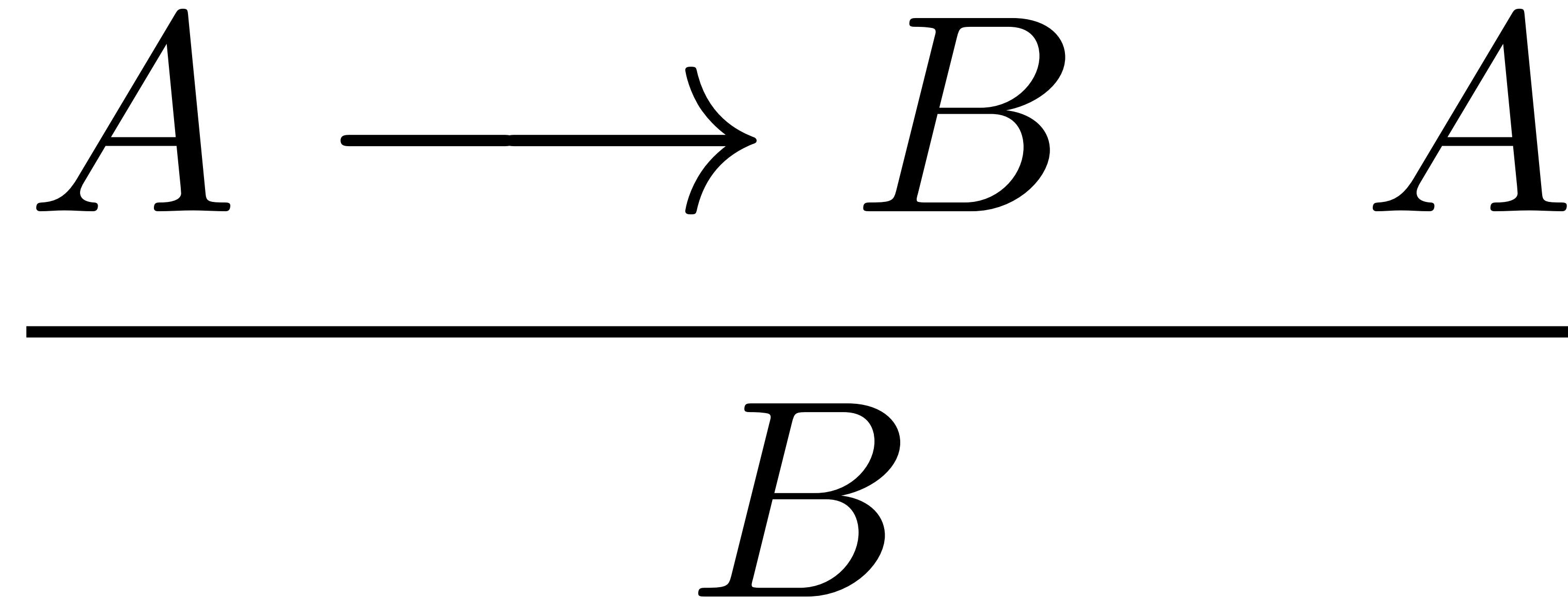
Gradiente do paradigma funcional em linguagens

- Linguagens Funcionais Puras (Haskell)
- Linguagens Funcionais com construções Imperativas (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



ML (OCaml) is logic in programming

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

Example in Java

B f1(A a) {...}

$$f1 : A \rightarrow B$$

C f2(B b) {...}

$$f2 : B \rightarrow C$$

D f3(C c) {...}

$$f3 : C \rightarrow D$$

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

$$\frac{\begin{array}{c} A \quad A \rightarrow B \\ \hline B \end{array}}{\begin{array}{c} B \quad B \rightarrow C \\ \hline C \end{array}} \quad \begin{array}{c} C \rightarrow D \\ \hline \end{array}$$

$$D$$



- É um dialecto da família ML
- É uma linguagem de expressões que:
 - Ou denotam um valor,
 - Ou terminam com uma exceção
 - Ou não terminam...
- É fortemente tipificada com inferência de tipos



Xavier Leroy



Didier Rémy



Damien Doligez



Jérôme Vouillon

facebook

 Microsoft

 docker

 Jane
Street

Bloomberg

ț

 ahrefs

Ambientes de programação

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



```
utop # 1. /. 2.  
;;  
- : float = 0.5
```

```
▶ ▾  
[3] 1+2*3/2  
    ✓ 0.0s  
... - : int = 4
```

Tipos básicos, literais, operadores e funções

| Tipo | Literal | Operador | Função |
|---------|------------------|-------------|-----------------------------|
| int | 1 2 3 0b101 0x42 | + - * / | |
| float | 1. 2. 3.5 4e10 | +. -. *. /. | float_of_int |
| bool | true false | && | |
| char | 'a' 'b' | | char_of_int, int_of_char |
| 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

| Tipo | Literal | Operador | Função |
|---------|------------------|-------------|-----------------------------|
| int | 1 2 3 0b101 0x42 | + - * / | |
| float | 1. 2. 3.5 4e10 | +. -. *. /. | float_of_int |
| bool | true false | && | |
| char | 'a' 'b' | | char_of_int, int_of_char |
| strings | "hello" "" | ^ | string_of_int |
| unit | () | | |

```
▶ 3.14 *. (float_of_int 2)
[5] ✓ 0.0s
... - : float = 6.28
```

Tipos básicos, literais, operadores e funções

| Tipo | Literal | Operador | Função |
|---------|------------------|-------------|-----------------------------|
| int | 1 2 3 0b101 0x42 | + - * / | |
| float | 1. 2. 3.5 4e10 | +. -. *. /. | float_of_int |
| bool | true false | && | |
| char | 'a' 'b' | | char_of_int, int_of_char |
| 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

| Tipo | Literal | Operador | Função |
|---------|------------------|-------------|-----------------------------|
| int | 1 2 3 0b101 0x42 | + - * / | |
| float | 1. 2. 3.5 4e10 | +. -. *. /. | float_of_int |
| bool | true false | && | |
| char | 'a' 'b' | | char_of_int, int_of_char |
| strings | "hello" "" | ^ | string_of_int |
| units | / \ | | |

```
▶ "abc".[0]
[7] ✓ 0.0s
... - : char = 'a'
```

Igual e Igual

- A igualdade em OCaml pode ter inúmeros sentidos, pode ser redefinida.
- No entanto, a linguagem permite duas formas nativas de testar a igualdade.
- Pode ser de forma **estrutural** onde dois valores têm a mesma estrutura e conteúdo.
- Ou então de forma **física**, onde dois valores apontam para o mesmo objeto em memória.

| Comparação | Estrutural | Física |
|--------------|-------------------|---------------------|
| Igualdade | = | == |
| Desigualdade | <> | != |
| Compara | Conteúdos | Posições de memória |
| Exemplo | [1;2;3] = [1;2;3] | [1;2;3] == [1;2;3] |

Asserções

- Asserções permitem o teste de uma expressão, lançando uma exceção no caso **falso**.

```
▶ 
  assert (int_of_string "42" = 43)
[8] ✘ 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
```

Célula da Kernel Número da linha Número da coluna

Expressões condicionais

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

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

- Note: o ramo `else` é obrigatório para todos os tipos excepto **unit**.

```
▶ ▾ if "hello" = "world" then 1
[10] ✘ 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
```