Linguagens e Ambientes de Programação (Aula Teórica 3)

LEI - Licenciatura em Engenharia Informática

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

NOVA SCHOOL OF
SCIENCE & TECHNOLOGY

Agenda para hoje

- Declaração de nomes, outra vez.
- Declaração de funções, com e sem parâmetros.
- Avaliação de expressões por substituição.
- Funções como valores (primeira vez).
- Avaliação parcial de funções

Declarações

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=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 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=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 importantes para a avaliação de uma expressão.

Âmbito das declarações locais

```
let x =
 let y = 1 in
 let z = 2 in
 y + z
in
 let w = 3+x in
 w + x
```

x

Âmbito das declarações locais

```
let x =
 let y = 1 in
 let z = 2 in
 y + z
in
 let w = 3+x in
 w + x
```

Y

Âmbito das declarações locais

```
let x =
 let y = 1 in
 let z = 2 in
 y + z                                    z
in
 let w = 3+x in
 w + x
```

Âmbito das declarações locais

```
let x =
 let y = 1 in
 let z = 2 in
 y + z
in
 let w = 3+x in
 w + x
```

w

Avaliação das declarações por substituição

```
let x =
  let y = 1 in
  let z = 2 in
  y + z
in
 let w = 3+x in
 w + x
```

Avaliação das declarações por substituição

```
let x =
 let y = 1 in
 let z = 2 in
 y + z
in
 let w = 3+x in
 w + x
```

Avaliação das declarações por substituição

```
let x =

  let z = 2 in
  1 + z
in
 let w = 3+x in
 w + x
```

Avaliação das declarações por substituição

```
let x =

 let z = 2 in
 1 + z
in
 let w = 3+x in
 w + x
```

Avaliação das declarações por substituição

```
let x =

 1 + 2
in
 let w = 3+x in
 w + x
```

Avaliação das declarações por substituição

```
let x =

 3
in
 let w = 3+x in
 w + x
```

Avaliação das declarações por substituição

```
let w = 3+3 in
w + 3
```

Avaliação das declarações por substituição

```
let w = 3+3 in
w + 3
```

Avaliação das declarações por substituição

```
let w = 6 in
w + 3
```

Avaliação das declarações por substituição

```
6 + 3
```

Avaliação das declarações por substituição

9

Funções

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)
- os parâmetros são listados na declaração.

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

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)
- os parâmetros são listados na declaração.
- as funções “sem parâmetros” têm um parâmetro de tipo unit.

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

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)
- os parâmetros são listados na declaração.
- a declaração com parâmetros é uma forma sintática alternativa à utilização de valores de tipo função (o símbolo seta é composto por dois caracteres ->)

```
let f = fun x → x+1 in f (1+1)
```

Definição e chamada (aplicação) de funções

- A aplicação de funções pode ser definida por substituição do parâmetro pelo valor do argumento.
- OCaml implementa uma estratégia de avaliação *call-by-value*, o que quer dizer que os argumentos são avaliados antes de expandir o corpo da função.

```
(fun x → x+1) (1+1)
(fun x → x+1) 2
2+1
3
```

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
```

```
(* [even x] is true if [x] is even, false otherwise
   Requires: [x ⩾ 0] *)
let rec even x = if x = 0 then true else if x = 1 then false else odd(x-1)

(* [odd x] is true if [x] is odd, false otherwise
   Requires: [x ⩾ 0] *)
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);
  }
}
```

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)
- os parâmetros são listados na declaração.

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

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)
- os parâmetros são listados na declaração.

```
let f = fun x y → x+y in f 1 1
```

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)
- os parâmetros são listados na declaração.

```
let f = fun x → fun y → x+y in f 1 1
```

Avaliação parcial de funções

- Funções com vários parâmetros são na realidade composição de várias funções.
- Os parâmetros podem ser instanciados um de cada vez resultando em funções que aceitam os restantes parâmetros até produzir um resultado final.

```
[9]  let add x y = x + y
     ✓ 0.0s
...  val add : int → int → int = <fun>

[10] add 2 3
     ✓ 0.0s
...  - : int = 5

[12] let add1 = add 1
     ✓ 0.0s
...  val add1 : int → int = <fun>

[13] add1 2
     ✓ 0.0s
...  - : int = 3
```

Sumário

- Declaração de nomes, outra vez.
- Declaração de funções, com e sem parâmetros.
- Avaliação de expressões por substituição.
- Funções como valores (primeira vez).
- Avaliação parcial de funções