Programação Funcional

Tipos Paramétricos

Estrutura de Linguagens
`https://github.com/fsantanna-uerj/EDL/`

Francisco Sant'Anna
`francisco@ime.uerj.br`

Tipos Paramétricos
Tipos Genéricos
Polimorfismo Paramétrico

- `map :: (a → b) → [a] → [b]`
  `f l m`
- `filter :: (a → Bool) → [a] → [a]`
  `f l m`
- `foldr :: (a → b → b) → b → [a] → b`
  `f` v_0 `l` v_n

Implementação

- listas
- map, filter, fold
- tipos polimórficos

Tipos Paramétricos

Tipos Algébricos

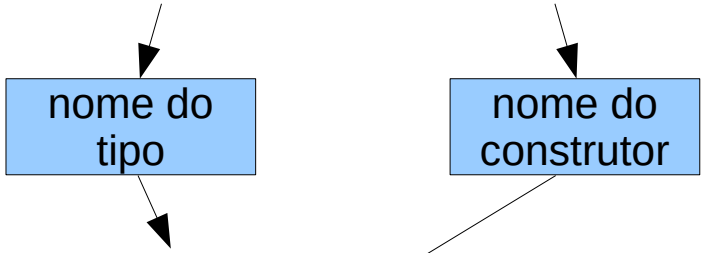
Estrutura de Linguagens
https://github.com/fsantanna-uerj/EDL/

Francisco Sant'Anna
francisco@ime.uerj.br

Tipos de Usuários

- **type** Pessoa = (**String**,**Int**,**Bool**)
  - O **type** define apenas um apelido (**typedef**)
  - p = (“Maria”, 20, False)

- **data** Pessoa = Pessoa String Int Bool



  - p :: Pessoa

    p = Pessoa “Maria” 20 False

data vs struct

```
data Pessoa = Pessoa String Int Bool
p :: Pessoa
p = Pessoa “Maria” 20 False
```

```
struct Pessoa {
   char nome[255];
   int  idade;
   bool masc;
};
struct Pessoa p = { “Maria”, 20, false };
```

data vs enum

- **data** Bool = False ou True


```
b :: Bool
b = False
```

- **enum** Bool { FALSE,TRUE };

```
enum Bool b = FALSE;
```

data vs enum/union/struct

- ```
  data Membro = Professor Int Int
              | Aluno Int Float
  ```

  OU

  ```
  m :: Membro
  m = Aluno 5 6.8
  ```

- ```
  enum MEMBRO { PROFESSOR,ALUNO };
  struct Membro {
    enum MEMBRO sub; // tagged union (uniao discriminada)
    union {
      struct { int alunos; int aulas; }; // PROFESSOR
  ```

  OU

  ```
      struct { int periodo; float cr; }; // ALUNO
    };
  };
  struct Membro m = { ALUNO, { 5, 6.8 } };
  ```

Exercícios

Considere um jogo estilo *RPG* com *Guerreiros*, *Magos* e *Sacerdotes*.

1. Crie um tipo de dados em Haskell que represente as três classes acima.
   - Considere que algumas propriedades são comuns às três classes (ex., nome, altura, idade, etc).
2. Crie o mesmo tipo de dados em C usando `enum`, `struct`, `union`.

Tipos Paramétricos

Tipos Algébricos

Estrutura de Linguagens
https://github.com/fsantanna-uerj/EDL/

Francisco Sant'Anna
francisco@ime.uerj.br

Tipos Paramétricos

Listas

Estrutura de Linguagens
https://github.com/fsantanna-uerj/EDL/

Francisco Sant'Anna
francisco@ime.uerj.br

Tipo Lista (não paramétrico)

```
l :: [Int]
l = [10, 20, 30]
l = 10 : 20 : 30 : []
```

```
data Lista = No Int Lista | Vazio
```

```
l = No 10 (No 20 (No 30 Vazio))
```

```
l = 10 `No` 20 `No` 30 `No` Vazio
```

```
struct no {
    int val;
    struct no* nxt;
};
```

Tipo Lista (paramétrico)

```
data ListaInt = No Int ListaInt | Vazio
l :: ListaInt
l = No 10 (No 20 (No 30 Vazio))
```

```
data Lista a = No a (Lista a) | Vazio
l :: Lista Int
l = No 10 (No 20 (No 30 Vazio))
m :: Lista Bool
m = No True (No False Vazio)
```

Exercícios

1. Considere a definição paramétrica de listas do slide anterior...
   - Crie uma lista que guarde sublistas de inteiros.

2. Crie um tipo de paramétrico para árvores binárias.
   1. Crie uma árvore binária de booleanos.
   2. Crie uma árvore em que cada nó guarda uma lista de inteiros.

Tipos Paramétricos

Listas

Estrutura de Linguagens
https://github.com/fsantanna-uerj/EDL/

Francisco Sant'Anna
francisco@ime.uerj.br

Tipos Paramétricos

Map

Estrutura de Linguagens

https://github.com/fsantanna-uerj/EDL/

Francisco Sant'Anna

francisco@ime.uerj.br

Map (não paramétrica)

```
data ListaInt = No Int ListaInt | Vazio

l = No 10 (No 20 (No 30 Vazio))

m = mapInt (*2) l

mapInt :: (Int → Int) → ListaInt → ListaInt

mapInt f l =
  case l of
    Vazio  → Vazio
    No x m → No (f x) (mapInt f m)
```

Map (paramétrica)

```
data Lista a = No a (Lista a) | Vazio

l = No 10 (No 20 (No 30 Vazio))

m = map (*2) l

map :: (a → b) → Lista a → Lista b

map f l =
  case l of
    Vazio  → Vazio
    No x m → No (f x) (map f m)
```

Exercícios

1. Implemente a função filter com a seguinte assinatura:
   - `filter :: (a → Bool) → [a] → [a]`

2. Implemente a função fold com a seguinte assinatura:
   - `fold :: (a → b → b) → b → [a] → b`

Tipos Paramétricos

Map

Estrutura de Linguagens

https://github.com/fsantanna-uerj/EDL/

Francisco Sant'Anna

francisco@ime.uerj.br

Tipos Paramétricos

Árvore Binária

Estrutura de Linguagens

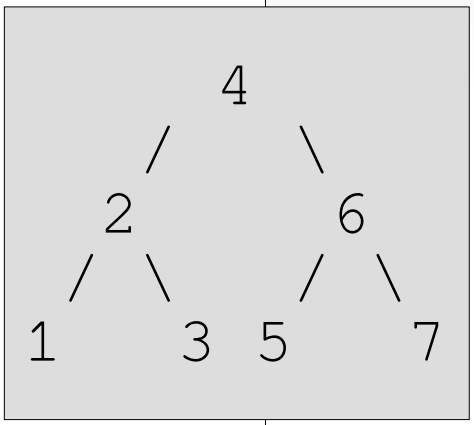
`https://github.com/fsantanna-uerj/EDL/`

Francisco Sant'Anna

`francisco@ime.uerj.br`

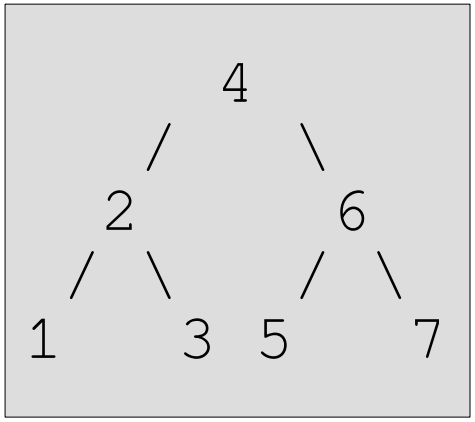
Árvore Binária

```
data Arvore a = Galho (Arvore a) a (Arvore a)
              | Folha

x :: Arvore Int
x = Galho (Galho (Galho Folha 1 Folha)
                 2
                 (Galho Folha 3 Folha))
          4
          (Galho (Galho Folha 5 Folha)
                 6
                 (Galho Folha 7 Folha))
```



Árvore Binária

```
data Arvore a = Galho (Arvore a) a (Arvore a)
              | Folha


tamanho :: Arvore a → Int
tamanho Folha = 0
tamanho (Galho e _ d) = 1 + tamanho e
                          + tamanho d
```



Exercícios

1. Implemente a função **`folhas`** com a seguinte assinatura:
   - `folhas :: Arvore a → Int`
2. Implemente a função **`mapA`** com a seguinte assinatura:
   - `mapA :: (a → b) → Arvore a → Arvore b`
3. Implemente a função **`lista`** com a seguinte assinatura:
   - `lista :: Arvore a → Lista a`

Tipos Paramétricos

Árvore Binária

Estrutura de Linguagens

`https://github.com/fsantanna-uerj/EDL/`

Francisco Sant'Anna

`francisco@ime.uerj.br`