# Tuplas

## Linguagem Haskell

Maria Adriana Vidigal de Lima

*Faculdade de Computação - UFU*

*Setembro - 2009*

# Tuplas e Listas

- A linguagem Haskell nos fornece dois mecanismos para a construção de dados compostos: **tuplas** e **listas**.
- Numa tupla podemos combinar os componentes de um dado numa única estrutura, e os componentes podem ter tipos e propriedades distintas.
- A lista possibilita a reunião de vários elementos - todos do mesmo tipo - numa única estrutura.

# Algumas Tuplas

```
(True, 1, 1)
("Alo Mundo", False)
(4, 5, "Seis", True, 'b')
((3,'a'), [1,2,3])
```

Uma tupla é uma seqüência ordenada de *n* elementos e possui as propriedades:

- pode conter mais de uma vez o mesmo elemento.
- os elementos são representados obrigatoriamente na ordem dada.

# Funções `fst` e `snd` para Duplas

As funções fst (first) e snd (second) podem ser utilizadas para recuperar os valores armazenados em duplas:

```
Prelude> fst (2, 5)
2
Prelude> fst (True, "Erro")
True
Prelude> snd (5, "Bom dia")
"Bom dia"
```

# Teste com `fst` e `snd`

As funções `fst` (first) e `snd` (second) podem ser usadas de forma combinada. Por exemplo, para recuperarmos o número 4 na tupla abaixo,

```
((True, 4),"Bom")
```

podemos escrever:

```
> snd (fst ((True, 4),"Bom") )
4
```

# Operações com tuplas

```
Main> (1,1) == (1,1)
True

Main> (1,1) < (2,2)
True

Main> (1,1,1) < (2,2)
ERROR - Type error in application

Main> (1,2,3) == (2,3,1)
False

Main> (1,2,3) == (,,)1 2 3
True
```

# Exemplo Biblioteca

No modelo dos dados de uma biblioteca, podemos representar cada objeto livro através dos atributos: *código*, *título*, *autor*, *editora*, *ano de publicação*.

Uma tupla permite a combinação desses atributos e o exemplo abaixo ilustra uma tupla representando um livro numa biblioteca:

```
("H123C9","Haskell","Thompson","Pearson",1999)
```

# Exemplo Biblioteca

As tuplas abaixo representam diferentes livros:

```
("H123C9","Haskell","Thompson","Pearson",1999)
("H214C5","Haskell","Sá","Novatec",2006)
("S612C1","SGBD","Ramakrishnan","McGraw-Hill",2008)
("L433C5","Linguagens","Tucker","McGraw-Hill",2009)
```

e os valores dos atributos (em cada livro) são do tipo:

```
(String, String, String, String, Int)
```

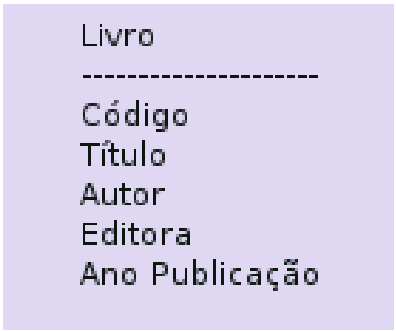
# Definição de Tipo

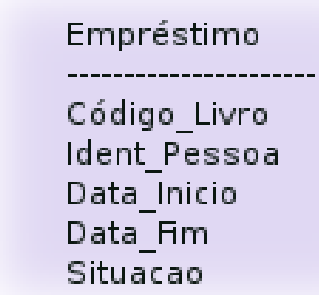
Podemos definir novos tipos de dados em Haskell, e cada tipo novo deve ser nomeado. Para o exemplo dos livros, podemos escrever:

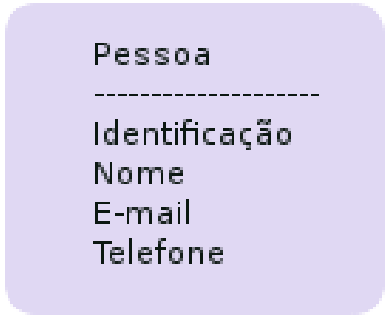
```
type Livro = (String, String, String, String, Int)

l1,l2::Livro
l1 = ("H123C9","Haskell","Thompson","Pearson",1999)
l2 = ("H214C5","Haskell","Sá","Novatec",2006)
```

# Exemplo Biblioteca




```
type Livro = (String, String, String, String, Int)
type Pessoa = (String, String, String, String)
type Emprestimo = (String, String, Data, Data, String)

type Data = (Int, Int, Int)
```

# Exemplo Biblioteca

```
type Data = (Int, Int, Int)
type Livro = (String, String, String, String, Int)
type Pessoa = (String, String, String, String)
type Emprestimo = (String, String, Data, Data, String)

l1::Livro
l1 = ("H123C9","Haskell","Thompson","Pearson",1999)

p1::Pessoa
p1 = ("BSI945","Ana Silva", "ana@email", "3322-1122")

e1::Emprestimo
e1 = ("H123C9","BSI200945",(12,9,2009),(20,9,2009),
      "aberto")
```

# Exemplo Biblioteca

```
type Livro = (String, String, String, String, Int)
type Livros = [Livro]

bdLivro::Livros
bdLivro =
  [("H123C9","Haskell","Thompson","Pearson",1999),
   ("H214C5","Haskell","Sá","Novatec",2006),
   ("S612C1","SGBD","Ramakrishnan","McGraw-Hill",2008),
   ("L433C5","Linguagens","Tucker","McGraw-Hill",2009)]
```

`bdLivro` é uma lista de livros (representados por tuplas)

# Exemplo Biblioteca

```
type Pessoa = (String, String, String, String)

type Pessoas = [Pessoa]

bdPessoa::Pessoas
bdPessoa =
  [("BSI945","Ana Silva", "ana@email", "3322-1122"),
   ("BCC021","Antonio Matos", "ant@email", "1122-1100"),
   ("BSI030","Augusto Melo", "aug@email", "1234-1234")]
```

`bdPessoa` é uma lista de pessoas (tuplas)

# Exemplo Biblioteca

```
type Emprestimo = (String, String, Data, Data, String)

type Emprestimos = [Emprestimo]

bdEmprestimo::Emprestimos
bdEmprestimo =
   [("H123C9","BSI945",(12,9,2009),(20,09,2009),
     "aberto"),
    ("L433C5","BCC021",(01,9,2009),(10,09,2009),
     "encerrado")]
```

`bdEmprestimo` é uma lista de empréstimos de livros para usuários.

# Exemplo Biblioteca

```
type Emprestimo = (String, String, Data, Data, String)

type Emprestimos = [Emprestimo]

bdEmprestimo::Emprestimos
bdEmprestimo =
   [("H123C9","BSI945",(12,9,2009),(20,09,2009),
     "aberto"),
    ("L433C5","BCC021",(01,9,2009),(10,09,2009),
     "encerrado")]
```

`bdEmprestimo` é uma lista de empréstimos de livros para usuários.

# Exercicio

Dada uma determinada data, verifique se é válida.

```
type Data = (Int,Int,Int)

bissexto:: Int-> Bool
bissexto x | (mod x 400 == 0) = True
           | (mod x 4 == 0) && (mod x 100 /= 0) = True
           | otherwise = False

valida::Data->Bool
valida (d,m,a)
   | d >= 1 && d <= 31 && (m == 1 || m == 3 || m == 5 ||
     m == 7 || m == 8 || m == 10 || m == 12) = True
   | d >= 1 && d <= 30 && (m == 4 || m == 6 || m == 9 ||
     m == 11) = True
   | d >= 1 && d <= 28 && m == 2 && not (bissexto a) = True
   | d >= 1 && d <= 29 && m == 2 && (bissexto a) = True
   | otherwise = False
```