Casamento de Padrões em Programação Funcional

Casamento de padrão

- **Casamento de padrão** é uma operação envolvendo um padrão e uma expressão que faz a correspondência (*casamento*) entre o padrão e o valor da expressão.
- O casamento de padrão pode suceder ou falhar, dependendo da forma do padrão e da expressão envolvidos.

Casamento de padrão (cont.)

- Em um casamento de padrão, o padrão e a expressão devem ser do mesmo tipo.
- Existem várias **formas de padrão**. Na sequência algumas delas serão apresentadas.

Padrão constante

O **padrão constante** é simplesmente uma constante.

O **casamento** sucede se e somente se o padrão for idêntico ao valor. Nenhuma associação de **variável** é produzida.

Exemplos:

padrão	valor	casamento
10	10	C
10	28	×
10	'P'	*erro de tipo*
'P'	'P'	C
'P'	'q'	×
'P'	**True**	*erro de tipo*
True	**True**	C
True	**False**	×
True	65	*erro de tipo*

C: sucede

×: falha

Padrão variável

O **padrão variável** é simplesmente um identificador de variável de valor (e como tal deve começar com letra minúscula). O **casamento** sucede sempre. A **variável** é associada ao valor.

Exemplos:

padrão	valor	casamento
`x`	`10`	C`x` ↣ `10`
`alfa`	`563.1223`	C`alfa` ↣ `563.1223`
`letra`	`'K'`	C`letra` ↣ `'K'`
`nomeCliente`	`"Ana Maria"`	C`nomeCliente` ↣ `"Ana Maria"`
`pessoa`	`("Ana",'F',16)`	C`pessoa` ↣ `("Ana",'F',16)`
`notas`	`[5.6,7.1,9.0]`	C`notas` ↣ `[5.6,7.1,9.0]`

Padrão curinga

- O **padrão curinga** é escrito como um sublinhado (_). O **casamento** sucede sempre.
- Nenhuma associação de **variável** é produzida.
- _ é também chamado de **variável anônima**, pois casa com qualquer valor sem dar nome ao valor.

Exemplos:

padrão	valor	casamento
_	`10`	C
_	`28`	C
_	`'P'`	C
_	`()`	C
_	`(18,3,2012)`	C
_	`"Ana Maria"`	C
_	`[5.6,7.1,9.0]`	C

Padrão tupla

Uma **tupla** de padrões também é um padrão

```
( padrao1, ..., padraon )
```

O **casamento** sucede se e somente se cada um dos padrões casar com o componente correspondente do valor.

Se as aridades do padrão tupla e do valor tupla forem diferentes, então ocorre um erro de tipo.

Padrão tupla(cont.)

Exemplos:

padrão	valor	casamento
`(18,True)`	`(18,True)`	C
`(97,True)`	`(18,True)`	×
`(18,False)`	`(18,True)`	×
`(18,'M')`	`(18,True)`	*erro de tipo*
`(18,True,'M')`	`(18,True)`	*erro de tipo*
`()`	`()`	C
`(x,y)`	`(5,9)`	C`x` ↦ `5`, `y` ↦ `9`
`(d,_,a)`	`(5,9,2012)`	C`d` ↦ `5`, `a` ↦ `2012`
`(x,y,z)`	`(5,9)`	*erro de tipo*
`(18,m,a)`	`(18,3,2012)`	C`m` ↦ `3`, `a` ↦ `2012`
`(d,5,a)`	`(18,3,2012)`	×
`(nome,sexo,_)`	`("Ana",'F',18)`	C`nome` ↦ `"Ana"`, `sexo` ↦ `'F'`
`(_,_,idade)`	`("Ana",'F',18)`	C`idade` ↦ `18`
`(_,(_,fam),9)`	`('F',("Ana","Dias"),9)`	C`fam` ↦ `"Dias"`
`(_,(_,fam),5)`	`('F',("Ana","Dias"),9)`	×

Definindo funções com casamento de padrão

Uma **definição de função** é formada por uma seqüência de equações.

Os **parâmetros** usados em uma equação para representar os argumentos são padrões.

Em uma **aplicação de função** o resultado é dado pela primeira equação cujos parâmetros casam com os respectivos argumentos, e cuja guarda (se houver) é verdadeira.

Se em todas as equações os casamentos de padrão falharem ou todas as guardas forem falsas, ocorre um erro de execução.

Geralmente o uso de padrões para especificar os argumentos torna a definição da função mais clara.

Definindo funções com casamento de padrão(cont.)

Exemplo:

```
Not :: Bool->Bool
not False=True
not True=False
```

A função `not` mapeia **False** a **True**, e **True** a

```
not False      ~ True
not(even 6)    ~ False
```

Definindo funções com casamento de padrão(cont.)

Exemplo:

```
(&&) :: Bool->Bool->Bool
True && True=True
True && False=False
False && True=False
False && False=False
```

```
True && True   ~ True
False && True  ~ False
2>3 && odd4    ~ False
```

Definindo funções com casamento de padrão(cont.)

Exemplo:

```
(&&) ::Bool->Bool->Bool
True  &&  True=True
_         &&  _=False
```

```
True  &&  True    ~ True
False  &&  True  ~ False
2>3  &&  2<3       ~ False
```

Definindo funções com casamento de padrão(cont.)

Exemplo:

```
(&&) ::Bool->Bool->Bool
True  && b  = b
False && _  = False
```

```
True && True   ~ True
2>3 && 2<3     ~ False
2<3 && even5   ~ False
```

Definindo funções com casamento de padrão(cont.)

Exemplo:

```
(&&) :: Bool->Bool->Bool
b && b = b
_ && _ = False
```

Está incorreto, pois não é possível usar uma variável mais de uma vez nos padrões (**princípio da linearidade**).

Definindo funções com casamento de padrão(cont.)

Exemplos:

```
fst(x, _) = x
```

```
snd( _,y) = y
```

```
fst(1+2,1-2)          ~ 3
snd(div 50,even 9)    ~ False
```

Exercícios

Exercício 1

Dê três possíveis definições para o operador lógico ou (||), utilizando casamento de padrão.

Exercícios(cont.)

Exercício 2

Redefina a seguinte versão do operador lógico e (&&) usando expressões condicionais ao invés de casamento de padrão:

```
True  &&  True=True
_ &&  _=False
```

Exercício 3

Redefina a seguinte versão do operador lógico e (&&) usando expressões condicionais ao invés de casamento de padrão:

```
True    &&b    = b
False  &&  _=False
```

Exercícios(cont.)

Exercício 4

Defina uma função que recebe dois pontos no espaço e retorna a distância entre eles. Considere que um ponto no espaço é representado por uma tripla de números que são as coordenadas do ponto. Use casamento de padrão.

Exercício 5

Analise a seguinte definição e apresente uma definição alternativa mais simples desta função.

```
opp :: (Int,(Int,Int)) ->Int
opp z  = if fst z==1
         then fst (snd z)+snd (snd z)
         else  if fst z==2
                 then fst (snd z)-snd (snd z)
                 else 0
```