

Testes

Programação Funcional
DCOMP/UFS

Funções corretas

- Como podemos estar seguros de que uma função calcula o que deveria calcular?
 - Testes
 - Provas
- Como testar?
 - Usando o GHCi, selecionamos algumas entradas e verificamos o resultado
 - Testes baseados em propriedades

```
paraMaiusc :: Char -> Char
paraMaiusc c
  | eMinusc c = toEnum (fromEnum c - fromEnum 'a' +
                        fromEnum 'A')
  | otherwise = c
```

usando o GHCi podemos conferir que

```
> paraMaius 'f'
'F'
> paraMaius '%'
'%'
```

Que valores de testes escolho?

Quantos?

Caixa preta: Não tenho acesso ao código

Caixa branca: conheço o código

testes devem exercitar todos os *branches*

- Caixa preta: Não vejo o código
 - Identificar grupos de testes
 - O que é relevante? Irrelevante?
 - Identificar casos especiais
 - O que ocorre nos limites dos grupos?
- Caixa branca: Vejo o código
 - Além dos princípios anteriores, testes devem exercitar todos os *branches*

Testes baseado em propriedades

- Ao invés de testar valores retornados por uma função, podemos focar em testar alguma(s) propriedade(s) da função;
- Exemplo: uma propriedade da função `paraMaiusc`
 - para qualquer `c`, o valor de `paraMaiusc c` nunca é uma letra minúscula
- Podemos expressar propriedades usando funções que retornam `Bool`

```
prop_paraMaiusc :: Char -> Bool
prop_paraMaiusc c = not (eMinus (paraMaiusc c))
```

A função pré-definida `quickCheck` permite testar propriedades gerando entradas aleatórias

```
*Main> quickCheck prop_praMaius  
+++ OK, passed 100 tests.
```

Outro exemplo

```
maxi :: Int -> Int -> Int
maxi m n
    | m >= n      = m
    | otherwise = n
```

Qual seria uma propriedade relevante?

```
prop_Maxi1 :: Int -> Int -> Bool
prop_Maxi1 m n = maxi m n >= m
```

```
prop_Maxi2 :: Int -> Int -> Bool
prop_Maxi2 m n = maxi m n >= m &&
                  maxi m n >= n
```

Alguma outra propriedade relevante de `maxi`?

```
prop_Maxi3 :: Int -> Int -> Bool
prop_Maxi3 m n = maxi m n == m ||
                 maxi m n == n
```

Observe que `prop_Maxi2` e `prop_Maxi3` definem precisamente o maior entre dois números

```
*Main> quickCheck prop_Maxi2
+++ OK, passed 100 tests.
*Main> quickCheck prop_Maxi3
+++ OK, passed 100 tests.
```


Suponha que erramos a definição

```
maxi :: Int -> Int -> Int
maxi m n
    | m >= n      = m
    | otherwise = m
```

```
*Main> quickCheck prop_Maxi3
+++ OK, passed 100 tests.
*Main> quickCheck prop_Maxi2
*** Failed! Falsifiable (after 3 tests and 1 shrink):
0
1
```

O que é preciso para poder usar o quickCheck?

1. instale o pacote quickCheck no seu computador. Abra um terminal e execute:

```
cabal update  
cabal install QuickCheck
```

2. No arquivo .hs com suas definições inclua

```
import Test.QuickCheck
```

Exercícios

- Defina propriedades para a função `MaisProximoDaMedia` e teste-as com `quickCheck`
- Usando guardas, defina funções para
 - Calcular o menor de três números inteiros
 - Dados três números, calcular quantos estão acima da média
 - Dados os coeficientes a , b e c de uma equação de segundo grau $ax^2+bx+c=0$

defina duas funções para calcular as raízes menor e maior.

- Para as funções acima, defina propriedades e teste-as com `quickCheck`