sumList [2,3,4,5]
= 2 + sumList [3,4,5]
= 2 + (3 + sumList [4,5])
= 2 + (3 + (4 + sumList [5]))
= 2 + (3 + (4 + (5 + sumList [])))
= 2 + (3 + (4 + (5 + 0)))
= 14

Programação Funcional

Estruturas Condicionais

Matheus Carvalho Viana
matheuscviana@ufsj.edu.br

Dcomp
Departamento de Ciência da Computação
UFSJ
Universidade Federal de São João del-Rei

Introdução

A capacidade de poder decidir entre executar um conjunto de ações ou outro, dada uma condição, é o que difere o computador de uma calculadora.

Assim como as linguagens imperativas, as linguagens funcionais dispõem de mecanismos para que seus programas realizem esse tipo de decisão.

No Haskell existem 4 formas de avaliar condições:

1. Casamento de padrões
2. Comando if-then-else
3. Guardas
4. Case

Casamento de Padrões

As entradas das funções podem ser definidas com entradas constantes e chama a versão correspondente da função quando há um casamento entre entrada constante e o valor passado na chamada.

```
fatorial :: Int -> Int

fatorial 0 = 1

fatorial n = n * fatorial (n-1)
```

Casamento de Padrões

O símbolo *underline* (_) pode ser usado para indicar as situações em que uma entrada não interfere no resultado.

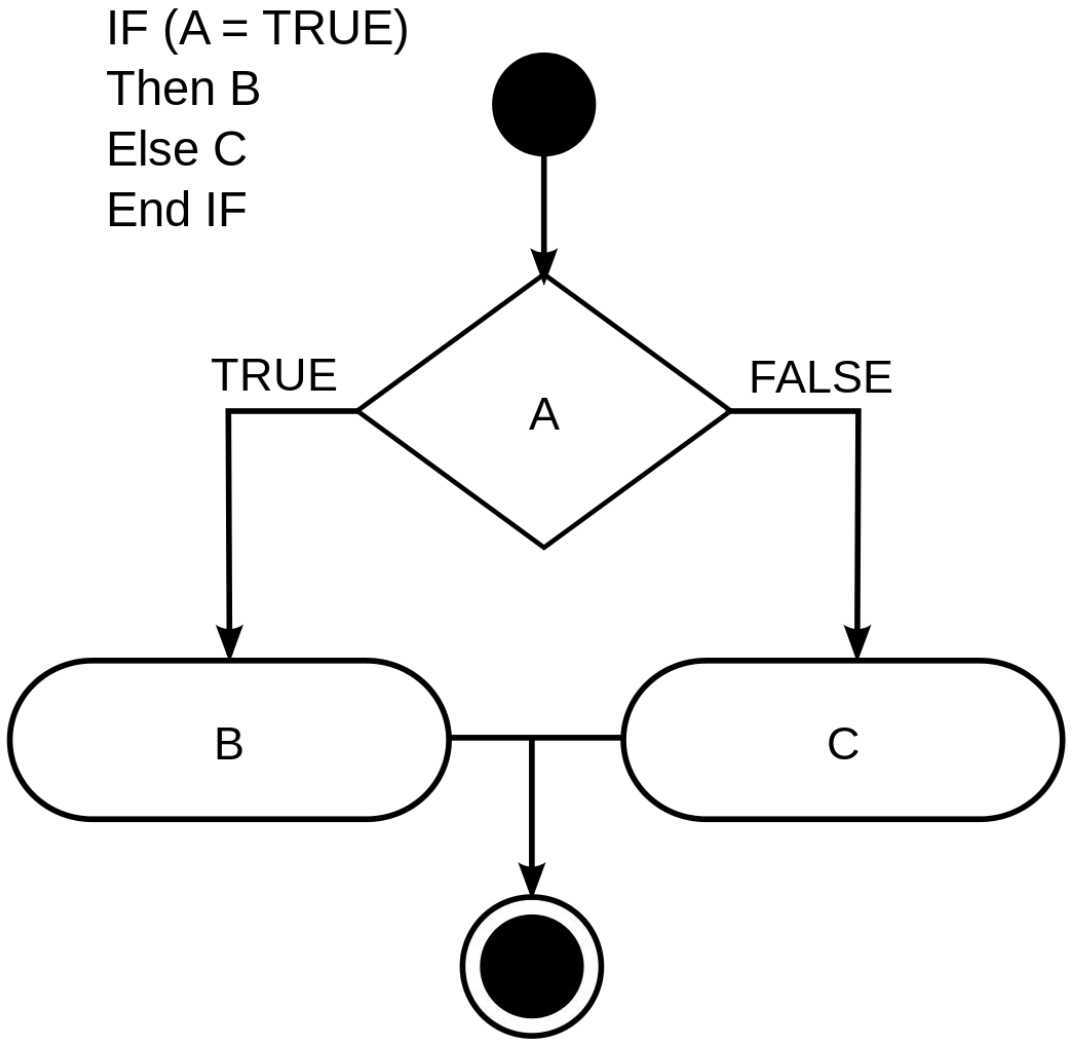
```
eLogico :: Bool -> Bool -> Bool
eLogico True True = True
eLogico False _ = False
eLogico _ False = False

funcao :: Double -> Double -> Double
funcao 0 _ = 1
funcao x y = (sin x) / x * y

fib :: Int -> Int
fib 1 = 0
fib 2 = 1
fib n = fib (n-2) + fib (n-1)
```

if-then-else

Expressão condicional de dois caminhos.

Sua sintaxe é:

```
if <condição>
  then <solução_True>
  else <solução_False>
```



if-then-else

```
maior :: Int -> Int -> Int
maior a b = if a > b then a else b

funcao :: Double -> Double -> Double
funcao x y = if x == 0 then 1 else (sin x) / x * y

fatorial :: Int -> Int
fatorial x = if x <= 1 then 1 else x * fatorial (x-1)

fib:: Int -> Int
fib n = if n == 1
  then 0
  else if n == 2 then 1 else fib (n-2) + fib (n-1)
```

Guardas

São uma forma de condicional de caminhos múltiplos.

Sua sintaxe é:

```
<função> <entradas>
  | <condição1> = <solução1>
  | <condição2> = <solução2>
  ...
  | <condiçãoN> = <soluçãoN>
  | otherwise = <solução_default>
```

Guardas

```
maior :: Int -> Int -> Int
maior a b
  | a > b = a
  | otherwise = b

funcao :: Double -> Double -> Double
funcao x y
  | x == 0 = 1
  | otherwise = (sin x) / x * y

fatorial :: Int -> Int
fatorial n
  | n == 0 = 1
  | n > 0 = n * fatorial (n-1)

fib :: Int -> Int
fib n
  | n == 1 = 0
  | n == 2 = 1
  | otherwise = fib (n-2) + fib (n-1)
```

Case

Outra forma de condicional de caminhos múltiplos.

Semelhante ao switch do C.

Sua sintaxe é:

```
case <variável> of
  <valor1> -> <solução1>
  <valor2> -> <solução2>

  <valorN> -> <soluçãoN>
  _ -> <solução_default>
```

Case

```
funcao :: Double -> Double -> Double
funcao x y = case x of
  0 -> 1
  _ -> (sin x) / x * y

fatorial :: Int -> Int
fatorial n = case n of
  0 -> 1
  _ -> n * fatorial (n-1)

fib :: Int -> Int
fib n = case n of
  1 -> 0
  2 -> 1
  _ -> fib (n-2) + fib (n-1)
```

Considerações

O casamento de padrões e o case são eficazes para os casos em que se quer definir uma solução para determinados valores de entrada específicos.

Entretanto essas estruturas não são úteis para os casos em que se quer definir uma solução para uma faixa de valores de entrada. Assim, as demais opções de condicionais devem ser aplicadas.

Exercícios

1. Crie uma função recursiva para dividir dois inteiros sem usar / ou div.
2. Faça uma função que recebe 3 números e retorna o maior.
3. Faça uma função que verifique se um ano é bissexto ou não. Um ano é bissexto se ele for divisível por 400 ou é divisível por 4 mas não por 100.
4. Faça uma função que solicite 3 inteiros e os imprime em ordem crescente. Use *show* para converter de inteiro para String.
5. Faça uma função que lê 2 inteiros e verifica se eles são múltiplos ou não.
6. Faça uma função que recebe um caractere e retorna True se for uma letra ou False, caso não seja.
7. Escreva um algoritmo que solicite a os três lados de um triângulo e verifique se é isóscele, equilátero ou escaleno.