Expressões condicionais

- Expressão condicional
- Definição de função com expressão condicional
- Equações com guardas
- Definições locais e guardas

Expressão condicional

- Expressões condicionais permitem a escolha entre duas alternativas na obtenção do valor da expressão,
- com base em uma condição (expressão lógica).
- exemplo: se-então-senão
- se delta == 0 então x1==x2 senão x1 != x2

Expressão condicional

- Em Haskell, uma expressão condicional tem a forma
- **if** *condição* **then** *exp1* **else** *exp2*
- if-then-else
- onde *condição* é uma expressão booleana (chamada predicado) e
- *exp1* (chamada consequência) e *exp2* (chamada alternativa) são expressões de um **mesmo tipo**.

Expressão condicional

- O valor da expressão condicional é o valor de exp1 se a condição é verdadeira, ou o valor de exp2 se a condição é falsa.
- Seguem alguns exemplos de expressões condicionais e seus valores.
- if True then 1 else 2

1

- if False then 1 else 2

2

Expressão condicional

- Seguem outros exemplos de expressões condicionais e seus valores.
- if 2 > 1 then "OK" else "FAIL"

“OK"

- if even 5 then 3+2 else 3-2

1

Expressão condicional

- A expressão condicional é uma expressão, e portanto sempre tem um valor.
- Assim uma expressão condicional pode ser usada dentro de outra expressão.

- 5 * (if True then 10 else 20)

?

Expressão condicional

- Veja os exemplos seguintes.
- 5 * (if True then 10 else 20)

50

- 5 * if True then 10 else 20

50

- length (if 2 <= 1 then "OK" else "FAIL")

4

Expressão condicional

- A cláusula **else** de uma expressão condicional não é opcional.
- Omiti-la é um erro de sintaxe.
- Se fosse possível omiti-la, qual seria o valor da expressão quando a condição fosse falsa?
  - Não teria nenhum valor neste caso, o que seria um problema.

Expressão condicional

- Assim uma expressão condicional sempre deve ter as duas opções (consequência e alternativa).
- Por exemplo, a seguinte expressão apresenta um erro de sintaxe, pois foi omitida a cláusula else.

- if True then 10

ERRO DE SINTAXE

Expressão condicional

- Regra de inferência de tipo
- test :: Bool
- e1 :: *a* - - *a* significa qualquer tipo válido
- e2 :: *a*
- if test then e1 else e2 :: ***a***

- Observe que a consequência e a alternativa devem ser do mesmo tipo, que também é o tipo do resultado.

Expressão condicional

- Exemplos no ambiente interativo:
- Prelude> :type if 4 > 5 then ’S’ else ’N’

if 4>5 then ’S’ else ’N’ :: **Char**

- Prelude> :type if mod 17 2 == 0 then length "banana" else 0

if mod 17 2 == 0 then length "banana" else 0 :: **Int**

Expressão condicional

- Exemplos no ambiente interativo:
- Prelude> if mod 17 2 /= 0 then not True else 'H'
  <interactive>:7:37:
- Couldn't match expected type 'Bool' with actual type 'Char'
- In the expression: 'H'
- In the expression: if mod 17 2 /= 0 then not True else 'H'
- In an equation for 'it':
- it = if mod 17 2 /= 0 then not True else 'H'

Definição de função com expressão condicional

- Como na maioria das linguagens de programação, funções podem ser definidas usando expressões condicionais.
- Na matemática, a função para calcular o valor absoluto de um número inteiro é definida como segue:
- $|n| = n \quad \text{se } n>=0$

  $-n \quad \text{se } n<0$
- Como seria em Haskell?

Definição de função com expressão condicional

- Em Haskell, a função para calcular o valor absoluto de um número inteiro pode ser definida como segue:
- valorAbsoluto :: Int -> Int
- valorAbsoluto n = if n >= 0 then n else -n
- valorAbsoluto recebe um inteiro n e resulta em n se ele é não-negativo, e -n caso contrário.

Definição de função com expressão condicional

- Expressões condicionais podem ser aninhadas, como mostra o exemplo a seguir, onde é definida uma função para determinar o sinal de um número inteiro.

- sinal :: Int -> Int
- sinal n = if n < 0
  - then -1
    else if n == 0
    - then 0
    - else 1

Equações com guardas

- Funções podem ser definidas através de equações com **guardas**,
- onde uma sequência de expressões lógicas, chamadas guardas,
- é usada para escolher entre vários possíveis resultados.
- Para evitar o aninhamento de expressões condicionais.

Equações com guardas

- Uma equação com guarda é formada por uma sequência de cláusulas escritas logo após a lista de argumentos.
- Cada cláusula é introduzida por uma barra vertical (|) e consiste em uma condição, chamada guarda,
- e uma expressão (resultado), separados por =

$$
\begin{aligned}
& f \;\; arg_1 \;\; \ldots \;\; arg_n \\
& \qquad\quad | \;\; guarda_1 \;\; = \;\; exp_1 \\
& \qquad\quad \vdots \\
& \qquad\quad | \;\; guarda_m \;\; = \;\; exp_m
\end{aligned}
$$

Equações com guardas

- Como exemplo, considere uma função para calcular o valor absoluto de um número:
- vabs :: Integer -> Integer
- vabs n

```
    | n >= 0 =  n
    | n < 0   = -n
```

Equações com guardas

- Observe que:
  - cada guarda deve ser uma **expressão lógica**, e
  - os resultados devem ser todos do **mesmo tipo**.
- Quando a função é aplicada, as guardas são verificadas na sequência em que foram escritas.
- A primeira guarda verdadeira define o resultado.

Equações com guardas

- Nesta definição da função de valor absoluto - vabs
- as guardas são:
  - n >= 0
  - n < 0
- e as expressões associadas são respectivamente
  - n
  - -n

Equações com guardas

- Veja um exemplo de aplicação da função:
- vabs 89
  ?? 89 >= 0

  ?? ==> True

==> 89

Equações com guardas

- Observe que quando o cálculo do valor de uma expressão é escrito passo a passo, indicamos o cálculo das guardas separadamente em linhas que começam com ??
- Veja outra exemplo de aplicação da função:
- vabs (75 - 2*50)
  ?? 75 - 2*50 >= 0

  ?? ==> 75 - 100 >= 0

  ?? ==> -25 >= 0
  ?? ==> False
  ?? -25 < 0
  ?? ==> True

==> - (-25)

==> 25

Equações com guardas

- Note que o argumento (75 - 2*50) é avaliado uma única vez, na primeira vez em que ele é necessário (para verificar a primeira guarda).
- O valor igual -25
- O seu valor não é recalculado quando o argumento é usado novamente na segunda guarda ou no resultado.

Equações com guardas

- Esta é uma característica da avaliação **lazy**:
  - Um argumento é avaliado somente se o seu valor for necessário,
  - e o seu valor é guardado caso ele seja necessário novamente.
- Logo um argumento nunca é avaliado mais de uma vez.

Equações com guardas

- Observe que na definição de vabs o teste n < 0 pode ser substituído pela constante True,
- pois ele somente será usado se o primeiro teste n >= 0 for falso, e se isto acontecer, com certeza n < 0 é verdadeiro:
- vabs n

```
    | n >= 0 = n

    | True = -n
```

Equações com guardas

- A condição True pode também ser escrita como otherwise:
- vabs n

```
    | n >= 0 = n
    | otherwise = -n
```

- otherwise é uma condição que captura todas as outras situações que ainda não foram consideradas.
- otherwise é definida no Prelúdio simplesmente como o valor verdadeiro:
- otherwise :: Bool
- otherwise = True

Equações com guardas

- Equações com guardas podem ser usadas para tornar definições que envolvem múltiplas condições mais fáceis de ler,
- como mostra o exemplo a seguir para determinar o sinal de um número inteiro:
- sinal :: Int —> Int
- sinal n

```
    | n < 0 = -1

    | n == 0 = 0

    | otherwise = 1
```

Equações com guardas

- Como outro exemplo temos uma função para analisar o índice de massa corporal (imc):
- analisaIMC :: Float -> String
- analisaIMC imc

  | imc <= 18.5 = “Você está abaixo do peso”

  | imc <= 25.0 = “Você parece normal”

  | imc <= 30.0 = “Você está gordo”

  | otherwise = “Você está muito gordo”

Equações com guardas

- Uma definição pode ser feita com várias equações.
- Se todas as guardas de uma equação forem falsas, a próxima equação é considerada.
- Se não houver uma próxima equação, ocorre um erro em tempo de execução.

Equações com guardas

- Por exemplo:
- funcaoTeste :: Int -> Int -> Int
- funcaoTeste x y

```
    | x > y = 1

    | x < y = -1
```

Equações com guardas

- Resultado:
- funcaoTeste 2 3

-1

- funcaoTeste 3 2

1

- funcaoTeste 2 2

ERRO

Definições locais e guardas

- Uma equação pode ter definições locais que são introduzidas na cláusula where.
- O escopo dos nomes definidos localmente restringe-se à menor equação contendo a cláusula where,
- incluindo as guardas (quando houver) e os resultados da equação.

Definições locais e guardas

- Veja um exemplo de definição local em uma equação com guardas:
- g x y

  | x <= 10 = x + b

  |x > 10 = x - b

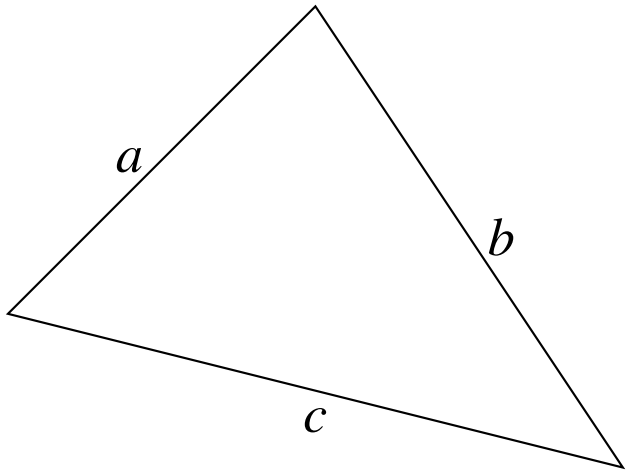
  where b = (y+1)^2
- O escopo de b inclui os dois possíveis resultados determinados pelas guardas.

Definições locais e guardas

- Outro exemplo, a função a seguir calcula a área de um triângulo somente quando as medidas dos lados do triângulo são válidas.
- Isto acontece somente quando cada medida é positiva e menor do que a soma das medidas dos outros dois lados.
- Caso as medidas não sejam válidas o resultado da função é zero.

Definições locais e guardas

- Utiliza-se a fórmula de Hieron para o cálculo da área A, sendo a, b e c as medidas dos lados, e s o semiperímetro do triângulo,



$$A = \sqrt{s(s-a)(s-b)(s-c)}$$

$$s = \frac{a+b+c}{2}$$

Definições locais e guardas

```
areaTriangulo a b c
    | medidasValidas = sqrt (s * (s-a) * (s-b) * (s-c))

    | otherwise = 0
    where

        medidasValidas = a > 0 && b > 0 && c > 0 &&

                                 a < b + c &&

                                 b < a + c &&

                                 c<a+b

        s = (a + b + c)/2
```

APS 4

- Crie o programa fonte aps4.hs (com todas as funções e variáveis) como codificado nesta aula. Teste todas as funções do programa fonte, carregando no GHCi.
- Observe a necessidade de definir os tipos de dados das variáveis e também das funções.
- Essa APS não precisa ser enviada para o professor, mas deve ser realizada.