Princípios de Programação

Trabalho 3

Universidade de Lisboa
Faculdade de Ciências
Departamento de Informática
Licenciatura em Engenharia Informática

2021/2022

A linguagem Haskell tem o seu avaliador de expressões aritméticas. Se baterem `3 + 1` no `ghci`, o interpretador calcula o valor e apresenta o resultado. Neste trabalho vamos definir e avaliar as nossas próprias expressões aritméticas. As expressões numéricas podem ter diversas utilizações, pelo que vamos construir um módulo com algumas funções para construir e avaliar expressões aritméticas.
O módulo deverá chamar-se `Expressoes` (sem til!) e deverá ser entregue num ficheiro de nome `t3_fcXXXXX.hs`, onde `XXXXX` é o seu número de aluno. Este módulo deverá exportar pelo menos os seguintes tipos de dados e funções:

1. O tipo de dados `Expr` mas não os seus construtores, de modo a que `Expr` seja um tipo de dados abstrato.

2. A função `constante :: Int -> Expr` que recebe um valor inteiro e devolve uma expressão que representa esse valor inteiro.

   ```
   ghci> :t constante 3
   constante 3 :: Expr
   ```

3. A função `variavel :: String -> Expr` que recebe um nome de uma variável e devolve uma expressão que representa essa variável.

   ```
   ghci> :t variavel "x"
   variavel "x" :: Expr
   ```

4. O operador `|+|`, com o tipo `(|+|) :: Expr -> Expr -> Expr`, que recebe duas expressões e devolve uma nova expressão que representa a soma das duas.

```
ghci> :t constante 3 |+| variavel "x"
constante 3 |+| variavel "x" :: Expr
```

5. O operador `|*|`, com o tipo `(|*|) :: Expr -> Expr -> Expr`, que recebe duas expressões e devolve uma nova expressão que representa o produto das duas.

```
ghci> :t constante 3 |*| variavel "y"
constante 3 |*| variavel "y" :: Expr
```

6. A função `avalia :: [(String,Int)] -> Expr -> Int`, que recebe uma lista de associação de variáveis a números inteiros, bem como uma expressão. Esta função realiza os cálculos necessários para devolver um inteiro, substituindo as variáveis pelos valores na lista de associação. Se uma variável não aparecer na lista, considere que o seu valor é 0.

```
ghci> avalia [] $ constante 4 |+| (constante 3 |*| variavel "x")
4
ghci> avalia [("x", 2)] $ constante 4 |+| (constante 3 |*| variavel "x")
10
ghci> avalia [("x", 2)] $ variavel "y" |+| (constante 3 |*| variavel "x")
6
```

   Note que até à chamada desta função, nenhuma operação é de facto realizada! As funções anteriores servem apenas para *construir* expressões.

7. Garanta que o tipo de dados `Expr` é instância da classe **`Eq`**. Para este trabalho, duas expressões são iguais quando os valores da sua avaliações são iguais, substituindo todas as variáveis por 0.

```
ghci> constante 4 |+| variavel "x" == constante 4
True
ghci> constante 4 |*| variavel "x" == constante 4
False
```

8. Garanta que o tipo de dados `Expr` é instância da classe **`Show`**. A representação textual de uma expressão é a sua representação em notação matemática comum. Deve utilizar sempre parêntesis e espaços em cada operador binário. Por exemplo:

```
ghci> constante 4 |+| (constante 3 |*| variavel "x")
(4 + (3 * x))
```

Notas

1. O seu trabalho é constituído pelo módulo `Expressoes`. Deverá submeter um ficheiro com o nome `t3_fcXXXXX.hs`, onde `XXXXX` é o seu número de aluno. O seu ficheiro deve começar por

   ```
   module Expressoes ( ... ) where
   ```

   onde no lugar de `...` deverão estar os vários tipos e funções pedidos neste enunciado.

2. Os trabalhos serão avaliados automaticamente. Respeite o nome do módulo, dos tipos de dados e das funções exportadas.

3. Cada função (ou expressão) que escrever deverá vir sempre acompanhada de uma assinatura. Isto é válido para as funções enunciadas acima bem como para outras funções ajudantes que decidir implementar.

4. Lembre-se que as boas práticas de programação Haskell apontam para a utilização de várias funções simples em lugar de uma função única mas complicada.

5. A precedência dos operadores não deve ser tomada em consideração. O uso de parêntesis é obrigatório em redor dos novos operadores.

Entrega. Este é um trabalho de resolução individual. Os trabalhos devem ser entregues no Moodle até às 23:55 do dia 22 de novembro de 2021.

Plágio. Os trabalhos de todos os alunos serão comparados por uma aplicação computacional. Relembramos aqui um excerto da sinopse: "Alunos detetados em situação de fraude ou plágio, plagiadores e plagiados, ficam reprovados à disciplina (sem prejuízo de ser acionado processo disciplinar concomitante)".