Tipos e Conversões

Cada linguagem tem restrições quanto a utilização de tipos. A linguagem C é um grande exemplo de "weakly typed". Conseguimos realizar operações com tipos diferentes simplesmente usando cast, conseguimos somar caracteres (uma operação normalmente criada para números), etc.

Haskell, por outro lado, é um exemplo de linguagem "strongly typed" e, na prática, isso é o oposto da permissividade de C. Nela, existem relações restritas entre tipos e regras fortes sobre como esses tipos devem se relacionar e se devem. Não é possível, por exemplo, somar dois caracteres. A operação 'a' + 'b' é algo que não faz o menor sentido em Haskell.

Se quiséssemos realmente somar dois caracteres, poderíamos fazê-lo com o auxílio da função fromEnum ("antônimo" de toEnum). Abaixo, a função sumCaracteres recebe dois caracteres, os converte e retorna a soma deles.

```
sumCaracteres :: Char -> Char -> Int
sumCaracteres x y = ( fromEnum x ) + ( fromEnum y )
```

Nesse mesmo contexto de conversão de tipos, dentre as tantas coisas que fazemos em Haskell, uma que aparece com frequência é a necessidade de converter inteiros em Strings e vice versa. Para a nossa sorte, Haskell já possui funções com esse propósito e elas são "show" e "read".

```
intToString :: Int -> String
intToString x = show x

stringToInt :: String -> Int
stringToInt x = read x
```

Poderíamos, por exemplo, criar uma função que soma dois valores a partir de duas strings.

```
sumStrings :: String -> String -> String
sumStrings x y = show xy
    where xy = xval + yval
          xval = read x
          yval = read y
```