# Aula Teórico-prática 9

Programação Funcional

LEI 1º ano (2006/2007)

Relembre da aula anterior os tipos.

```
data ExpInt = Const Int
            | Simetrico ExpInt
            | Mais ExpInt ExpInt
            | Menos ExpInt ExpInt
            | Mult ExpInt ExpInt

type ExpN = [Parcela]
type Parcela = [Int]
```

juntamente com as funções

```
calcula :: ExpInt -> Int
expString :: Exp -> String
posfix::Exp -> String
calcN :: ExpN -> Int
normaliza :: ExpInt -> ExpN
```

Uma possível generalização será considerar expressões cujas constantes de um qualquer tipo numérico (i.e., da classe `Num`). A definição desses tipos será agora

```
data Exp a = Const Int
           | Simetrico ExpInt
           | Mais ExpInt ExpInt
           | Menos ExpInt ExpInt
           | Mult ExpInt ExpInt

type ExpN a = [Parcela a]
type Parcela a = [a]
```

- Adapte as definições que apresentou das funções referidas a estes novos tipos.
- Sabendo o seguinte como *output* do `ghc`

  ```
  Prelude> :i Num
  class (Eq a, Show a) => Num a where
    (+) :: a -> a -> a
    (*) :: a -> a -> a
    (-) :: a -> a -> a
    negate :: a -> a
    abs :: a -> a
    signum :: a -> a
    fromInteger :: Integer -> a
  ```

  complete a seguinte definição:

  ```
  instance (Num a) => Num (Exp a) where
    .......
  ```

  Note que, em rigor, deverá ainda definir o tipo `Exp a` como uma instância de `Show` e de `Eq`.