Faculdade de Computação
Programação Funcional (BCC/BSI) - 1° Período
Aula Prática: Classes e Tipos Algébricos

Tipos Algébricos

Podemos definir em Haskell modelos para descrever novos tipos de dados usando:

- tipos básicos: Int, Float, Bool, Char
- tipos compostos: tuplas (t1,t2,...,tn); listas [t1], funções (t1 → t2) onde t1, t2, … tn são tipos.

Além dessas definições, podemos ainda escrever tipos enumerados como o tipo *estações do ano,* usando a palavra reservada *data:*

```
data EstacaoAno = Verao | Outono | Inverno | Primavera
```

O tipo `EstacaoAno` é definido por 4 construtores constantes: Verao, Outono, Inverno e Primavera, que serão os únicos valores possíveis para esse tipo.

Num tipo algébrico, os valores possíveis são definidos como construtores do tipo. O nome do tipo e o nome dos construtores deve começar sempre em letra maiúscula. Podem-se definir funções sobre estes tipos:

```
data EstacaoAno = Verao | Outono | Inverno | Primavera
data Temperatura = Calor |Frio

clima::EstacaoAno -> Temperatura
clima Inverno = Frio
clima _ = Calor
```

A função *clima* define que a temperatura será baixa quando for inverno, e alta em todas as outras estações do ano.

Para que duas estações do ano sejam comparáveis, podemos incluir o tipo `EstacaoAno` na classe de igualdade (Eq):

```
data EstacaoAno = Verao | Outono | Inverno | Primavera deriving (Eq)
```

Esta inclusão permite que duas estações do ano possam ser comparadas:

```
Verao == Inverno  = False
Verao == Verao    = True
```

Os construtores de um tipo algébrico podem representar uma enumeração, um conjunto de alternativas ou podem ser combinados gerando um tipo produto.

Exemplo de alternativas:

```
data Forma = Circulo Float | Retangulo Float Float
```

Exemplo de produto:

```
data NomePessoa = Nome String SobreNome String
```

Exercícios:

1) Seja a definição de tipo algébrico para modelar as horas:

```
data Hora =   AM Int Int
            | PM Int Int
```

Os valores do tipo Hora são escritos na forma `(AM x y)` ou `(PM x y)`, sendo `x` e `y` valores do tipo Int.

a) Teste as funções que retornam o total de horas, minutos e segundos de uma determinada hora:

```
totalHoras :: Hora -> Int
totalHoras (AM h m) = h
totalHoras (PM h m) = h+12

totalMinutos :: Hora -> Int
totalMinutos (AM h m) = h*60 + m
totalMinutos (PM h m) = (h+12)*60 + m

totalSegundos :: Hora -> Int
totalSegundos (AM h m) = h*3060 + m*60
totalSegundos (PM h m) = (h+12)*3600 + m*60
```

b) Modifique a função totalHoras para que sejam rejeitados valores para a variável h que estejam fora do intervalo de 0 a 11.

c) Faça os testes abaixo e explique os resultados:

```
> (AM 10 3)
> (AM 10 3) == (AM 10 3)
> (AM 10 3) > (PM 5 3)
```

d) Modifique a declaração para o tipo Hora de forma que os testes do item (c) possam ser bem sucedidos.

2) Sejam as seguintes declarações de tipo para modelar os dados dos alunos de uma turma. As disciplinas possíveis são Programação Funcional e Introdução à Programação de Computadores. As avaliações em cada disciplina serão feitas em duas etapas: Teoria e Prática e em ambos os casos o resultado da avaliação será um valor entre 0 e 50.

```
type Nome = [Char]
type Numero = Integer

data Curso = PF | IPC deriving (Eq,Show)

data Nota = Teoria Float | Pratica Float  deriving (Eq,Show)

type NotaAluno = (Nome, Numero, Curso, Nota)
type NotasTurma = [NotaAluno]
```

a) Crie uma lista de notas de alunos (tipo NotasTurma) contendo duas notas (Teoria e Prática) para cada aluno, considerando 5 alunos.

b) Defina um tipo algébrico para o Resultado, sendo que dois valores serão possíveis: Aprovado e Reprovado.

c) Implemente uma função que dado um aluno e uma lista de notas de alunos, verifique se o mesmo está Aprovado ou Reprovado, considerando que para ser aprovado o aluno precisa de uma nota superior ou igual a 60.

```
> verificaResultado "Ana" listaNotas
Aprovado
```

3) Defina um tipo algébrico para especificar valores monetários em Real, Dólar ou Euro, de forma que os valores (do tipo Float) possam ser diretamente comparados (com as devidas conversões), exibidos e ordenados.

4) Sejam as seguintes declarações de tipo para modelar três fusos horários usando o padrão UTC (Universal Time Zone).

> *A determinação dos fusos é determinada pelo meridiano 0 na cidade de Greenwich (UTC0), próxima a Londres. Desta forma, de quinze e quinze graus à leste, os fusos são numerados positivamente (+1, +2, +3...+12) e a oeste negativamente (-1, -2, -3...-12) .*

Os fusos UTC0, UTC+1 e UTC-1 podem ser definidos pelo tipo algébrico Fuso, e a Hora pode ser definida por uma tupla-3:

```
data Fuso = UTC0 | UTC1 | UTCm1 deriving (Eq)
type Hora = (Int,Int,Int)
```

a) Explique a função horaCidades e faça testes anotando os resultados.

b) Modifique o tipo Fuso e a função horaCidades incluindo novos fusos horários.

```
cidades = [("Londres",UTC0),("Lisboa",UTC0), ("Paris",UTC1),
           ("Roma", UTC1), ("Acores", UTCm1)]

horaCidades::(Fuso,Hora)->[(String,Fuso)]->[(String,Hora)]
horaCidades (UTC0, (h,m,s)) [] = []
horaCidades (UTC0, (h,m,s)) ((c,f):ls)
    | f == UTC1 =
          if h == 23 then (c,(0,m,s)): horaCidades (UTC0, (h,m,s)) ls
                     else (c,(h+1,m,s)): horaCidades (UTC0, (h,m,s)) ls
    | f == UTCm1 =
          if h == 0 then  (c,(23,m,s)): horaCidades (UTC0, (h,m,s)) ls
                    else (c,(h-1,m,s)): horaCidades (UTC0, (h,m,s)) ls
    | f == UTC0 = (c,(h,m,s)): horaCidades (UTC0, (h,m,s)) ls
```

Exercícios Adicionais:

5) Seja o programa abaixo para conversão de Valores representando Temperaturas nas escalas Celsius, Fahrenheit e Kelvin. Teste o programa e comente o código.

```
data Temperatura = Celsius Float | Fahrenheit Float
                 | Kelvin Float

emC::Temperatura->Temperatura
emC (Celsius c) = Celsius c
emC (Fahrenheit c) = Celsius ((c - 32) * 5 / 9)
emC (Kelvin c) = Celsius (c - 273)

instance Eq Temperatura where
    (Celsius c) == (Celsius d) = c == d
    t1 == t2 = (emC t1) == (emC t2)

instance Show Temperatura where
   show (Celsius c) = show c ++ "C"
   show (Fahrenheit c) = show c ++ "F"
   show (Kelvin c) = show c ++ "K"

instance Ord Temperatura where
  (Celsius x) > (Celsius y) = x > y
  t1 > t2 = (emC t1) > (emC t2)
  (Celsius x) < (Celsius y) = x < y
  t1 < t2 = (emC t1) < (emC t2)
  (Celsius x) >= (Celsius y) = x >= y
  t1 >= t2 = (emC t1) >= (emC t2)
  (Celsius x) <= (Celsius y) = x <= y
  t1 <= t2 = (emC t1) <= (emC t2)

emOrdem::[Temperatura]->Bool
emOrdem [a] = True
emOrdem (a:b:as) = if (a <= b) then emOrdem (b:as)
                               else False
```

6) Na escala Celsius, a temperatura de fusão do gelo é 0º C e da ebulição da água é de 100º C. Faça uma função que retorne numa tupla-2 duas listas: a primeira contendo os valores das temperaturas de fusão e a segunda de ebulição da água (nas escalas Celsius, Fahrenheit e Kelvin, respectivamente).

$$\frac{Tc}{5} = \frac{Tf - 32}{9} = \frac{Tk - 273}{5}$$

7) Use o Sistema Hugs para responder às perguntas abaixo e mostre o resultado:

a) Um corpo sofre um aquecimento de 40°C. Se este aquecimento fosse acompanhado pela escala Fahrenheit qual seria a variação nesta escala?

b) Um corpo sofre um aquecimento de 40°C. Se este aquecimento fosse acompanhado pela escala Kelvin qual seria a variação nesta escala?

8) Numa escala **TempX**, as convenções são 5°X para o ponto de fusão e 85°X para o ponto de ebulição da água. Para converter a leitura C (Celsius) em leitura X (TempX) ou o inverso temos:

$$X \rightarrow C: \ 5/4\ (X-5)$$
$$C \rightarrow X: \ 4/5\ C + 5$$

Inclua a convenção TempX no tipo algébrico Temperatura e implemente a conversão para Celsius.

9) Dada uma lista de temperaturas, em quaisquer escalas (C, F, K ou X), faça uma função que retorne uma tupla-2 contendo as temperaturas extremas (a menor e a maior). Exemplo:

```
> retornaExtremos [(Celsius 40.6), (Fahrenheit 89.9), (Kelvin 324),
(Celsius 13), (TempX 56), (Kelvin 112)]
(112.0°K,56.0°X)
```