Wed Jul 28 23:25:52 2021 Diogo Pinheiro Almeida (up202006059)
Logout

Problema 3: Análise de ficheiros *log*

Algo correu mal no *datacenter* do DCC e precisamos da vossa ajuda para perceber o que se passou. Felizmente temos os registos dos acontecimentos (*logs*). São ficheiros de texto com uma mensagem por linha; cada linha começa com uma letra indicando o tipo de mensagem:

- `I` para mensagens informativas;
- `W` para mensagens de aviso (*warnings*);
- `E` para mensagens de erro.

As mensagens de erros têm também um nível inteiro entre 1 e 100 que indica a gravidade da ocorrência (1 é o menos grave e 100 o mais grave). Todas as mensagens têm um inteiro que representa o tempo da ocorrência (*timestamp*); o resto da linha é o conteúdo da mensagem.

Eis um extrato de duas linhas com uma mensagem informativa no instante $t = 147$ seguida de um erro de nível 2 em $t = 148$:

```
I 147 mice in the air, I'm afraid, but you might catch a bat, and
E 2 148 #56k istereadeat lo d200ff] BOOTMEM
```

O ficheiro `sample.log` contém um pequeno exemplo destas mensagens; o ficheiro `error.log` contém todas as mensagens recuperadas do *datacenter*. Como este último ficheiro é longo, vamos escrever um programa para auxiliar a filtrar a informação. Começamos pelas declarações de tipos para estruturar a informação:

```haskell
-- tipos de mensagens
data MessageType = Info
                 | Warning
                 | Error Int   -- argumento: nível do erro
                deriving (Show, Eq)

type TimeStamp = Int   -- instante de tempo

-- entradas num ficheiro *log*
data LogEntry = LogMessage MessageType TimeStamp String
              | Unknown String
             deriving (Show, Eq)
```

formatadas e `unknown` representa outras linhas de texto que não sigam o formato indicado acima.

Preparação

Deve descarregar o ficheiro `Log.hs` com as declarações acima e colocá-lo no mesmo diretório em que vai desenvolver o seu código (num outro módulo). Para usar as definições do módulo `Log.hs` no seu programa deve colocar a seguinte declaração

```
import Log
```

no início do seu módulo. Não coloque as suas definições no módulo `Log.hs` porque não vai submeter esse ficheiro!

Exercício 1

Vamos começar por definir uma função para analisar uma linha de texto e converter num valor `LogEntry` apropriado.

- Exercício 1: Análise de uma mensagem
  (6 submissões)

Depois de fazer a análise de uma linha, podemos definir uma função para analisar o ficheiro completo usando `lines` para partir o conteúdo do ficheiro em linhas:

```
parseLog :: String -> [LogEntry]
parseLog txt = map parseMessage (lines txt)
```

Exercício 2

Agora que conseguimos fazer análise do ficheiro de *log* deparamos com um problema: as entradas estão fora de ordem! Vamos corrigir isso colocando as mensagens numa estrutura de árvore de pesquisa.

- Exercício 2: Inserir uma entrada numa árvore de pesquisa
  (4 submissões)

Exercício 3

Vamos usar a função anterior para colocar as mensagens por ordem.

- Exercício 3: Colocar mensagens por ordem
  (1 submissões)

Conclusão

Será que consegue juntar todas as componentes anteriores para descobrir o que aconteceu no incidente do *datacenter*? Escreva um programa que:

1. leia o ficheiro de texto `error.log` usando `readFile`;

(Este último programa não é para submeter.)

Adaptado de um exercício do curso CIS 194, University of Pensilvania.

Pedro Vasconcelos, 2021