

Trabalho prático TP1 – Etapa1

Programação Funcional - PL6
Outubro 2025

Docentes:
Arsénio Reis
Eduardo Nunes

Diogo Fernandes, al2024150129
Martim Fernandes, al2024146601
Rafael Gomes, al2024150362

Conteúdo

1. Introdução	2
2. Objetivos	2
Etapa 1	3
3. Estrutura de Dados	3
3.1 Tipos Definidos.....	3
3.2 Análise das Estruturas	4
4. Especificação das Funções	5
4.1 Funções de Leitura	5
4.2 Funções de Impressão.....	7
4.3 Função Principal.....	9
Contribuições:	10

1. Introdução

Este relatório documenta a implementação de um sistema de gestão de torneios desportivos desenvolvido em **Haskell**, uma linguagem de programação funcional pura.

O sistema foi concebido para gerir dois tipos distintos de competições:

- Torneios AVE (Todos Contra Todos): onde todas as equipas se defrontam entre si
- Torneios Eliminatórios: formato de eliminação direta com múltiplas fases

Contexto do Problema

- A gestão manual de torneios desportivos apresenta desafios significativos:
- Dificuldade em manter registos organizados de jogos e resultados
- Risco de erros na atualização de pontuações
- Falta de histórico estruturado para análise futura
- Ausência de ferramentas automatizadas para cálculo de classificações

Solução Proposta

- O sistema implementado oferece funcionalidades para:
- Leitura e processamento de dados de torneios a partir de ficheiros CSV
- Armazenamento estruturado de informações de jogos e resultados
- Visualização formatada de torneios e classificações
- Suporte para múltiplos formatos de competição

2. Objetivos

Os principais **objetivos** desta **primeira etapa** são:

1. **Leitura de Dados:** Implementar funções para importar dados de torneios a partir de ficheiros CSV
2. **Processamento:** Transformar dados brutos em estruturas organizadas
3. **Visualização:** Apresentar informações de forma clara e formatada
4. **Extensibilidade:** Permitir fácil adição de novos formatos de torneio

Etapa 1

3. Estrutura de Dados

3.1 Tipos Definidos

TorneioAVE

```
type TorneioAVE = [(String, Int, Int, String)]
```

Representa um torneio no formato "todos contra todos". Cada elemento da lista é um tuplo com:

- **String**: Nome da primeira equipa
- **Int**: Pontos marcados pela primeira equipa
- **Int**: Pontos marcados pela segunda equipa
- **String**: Nome da segunda equipa

Exemplo:

```
return [("Benfica", 2, 1, "Porto"), ("Sporting", 3, 0, "Braga")]
```

Justificação da Estrutura:

- Simplicidade: Tuplos são diretos e eficientes
- Imutabilidade: Segue os princípios funcionais
- Type-safe: Garante que cada jogo tem exatamente 4 componentes

ResultadosAVE

```
type ResultadosAVE = [(String, Int, Int, String, String)]
```

Estende TorneioAVE com informação adicional sobre o resultado:

- Primeiros 4 campos idênticos a TorneioAVE
- **String**: Resultado final ("Vitoria", "Empate", "Derrota")

Exemplo:

```
[("Benfica", 2, 1, "Porto", "Vitoria")]
```

TorneioElim

```
type TorneioElim = [(String, String, String)]
```

Representa torneios eliminatórios. Cada tuplo contém:

- **String:** Fase do torneio ("Oitavos", "Quartos", "Meias", "Final")
- **String:** Nome da primeira equipa
- **String:** Nome da segunda equipa

Exemplo:

```
[("Oitavos", "Benfica", "Porto"), ("Oitavos", "Sporting", "Braga")]
```

ResultadosElim:

```
type ResultadosElim = [(String, Int, Int, String)]
```

Armazena resultados de jogos eliminatórios:

- **String:** Fase do torneio
- **Int:** Pontos da primeira equipa (inferida do contexto)
- **Int:** Pontos da segunda equipa
- **String:** Nome da equipa derrotada

Exemplos:

```
[("Oitavos", 2, 1, "Porto"),
```

- Significa que em Oitavos, uma equipa venceu 2-1 o Porto

3.2 Análise das Estruturas

Vantagens

- **Simplicidade:** Tuplos são fáceis de entender e manipular
- **Eficiência:** Estruturas nativas de Haskell são otimizadas

Limitações Identificadas

- **Rigidez:** Difícil adicionar novos campos sem quebrar código existente
- **Ambiguidade:** Em ResultadosElim, não é claro qual equipa venceu

4. Especificação das Funções

4.1 Funções de Leitura

readTorneioAVE

CAMPO	DESCRIÇÃO
Nome	readTorneioAVE
Tipo	String -> IO TorneioAVE
Descrição	Lê dados de um torneio AVE a partir de um ficheiro CSV e retorna uma estrutura TorneioAVE
Parâmetros	filename - Caminho para o ficheiro CSV

Implementação Atual:

```
readTorneioAVE :: String -> IO TorneioAVE
readTorneioAVE filename = do
    conteudo <- readFile filename
    putStrLn "Conteúdo do ficheiro Torneio AVE: "
    putStrLn conteudo
    return [("Benfica", 2, 1, "Porto"), ("Sporting", 3, 0, "Braga")]
```

readResultadosTorneioAVE

CAMPO	DESCRIÇÃO
Nome	readResultadosTorneioAVE
Tipo	String -> IO ResultadosAVE
Descrição	Lê resultados de jogos do torneio AVE com classificação (vitória/empate/derrota)
Parâmetros	filename - Caminho para o ficheiro CSV de resultados

Implementação Atual:

```
readResultadosTorneioAVE :: String -> IO ResultadosAVE
readResultadosTorneioAVE filename = do
    conteudo <- readFile filename
    putStrLn "Conteúdo do ficheiro Resultados AVE: "
    putStrLn conteudo
    return [("Benfica", 2, 1, "Porto", "Vitoria"), ("Sporting", 3, 0, "Braga", "Vitoria")]
```

readTorneioElim

CAMPO	DESCRIÇÃO
Nome	readTorneioElim
Tipo	String -> IO TorneioElim
Descrição	Lê estrutura de um torneio eliminatório com fases e confrontos
Parâmetros	filename - Caminho para ficheiro CSV do torneio

Implementação atual:

```
readTorneioElim :: String -> IO TorneioElim
readTorneioElim filename = do
    conteudo <- readFile filename
    putStrLn "Conteúdo do ficheiro Torneio Eliminatório: "
    putStrLn conteudo
    return [("Oitavos", "Benfica", "Porto"), ("Oitavos", "Sporting", "Braga")]
```

readResultadosTorneioElim

CAMPOS	DESCRIÇÃO
Nome	readResultadosTorneioElim
Tipo	String -> IO ResultadosElim
Descrição	Lê resultados dos jogos eliminatórios
Parâmetros	filename - Caminho para ficheiro de resultados

Implementação atual:

```
readResultadosTorneioElim :: String -> IO ResultadosElim
readResultadosTorneioElim filename = do
    conteudo <- readFile filename
    putStrLn "Conteúdo do ficheiro Resultados Eliminatório: "
    putStrLn conteudo
    return [("Oitavos", 2, 1, "Porto"), ("Oitavos", 3, 0, "Braga")]
```

4.2 Funções de Impressão

printTorneioAVE

CAMPO	DESCRIÇÃO
Nome	printTorneioAVE
Tipo	TorneioAVE->IO
Descrição	Imprime torneio AVE em formato tabular legível
Parâmetros	xs - Lista de jogos do torneio

Implementação Atual:

```
printTorneioAVE :: TorneioAVE -> IO ()
printTorneioAVE xs = do
    putStrLn "Torneio AVE - Vila Real"
    putStrLn "====="
    putStrLn "Equipas,Pontos1,Equipas,Pontos2"
    putStrLn "Benfica,2,1,Porto"
    putStrLn "Sporting,3,0,Braga"
```

printResultadosTorneioAVE

CAMPOS	DESCRIÇÃO
Nome	printResultadosTorneioAVE
Tipo	ResultadosAVE -> IO ()
Descrição	Imprime resultados do torneio AVE incluindo classificação dos jogos
Parâmetros	xs - Lista de resultados

Implementação atual:

```
printResultadosTorneioAVE :: ResultadosAVE -> IO ()
printResultadosTorneioAVE xs = do
    putStrLn "Resultados Torneio AVE - Vila Real"
    putStrLn "====="
    putStrLn "Equipas,Pontos1,Equipas,Pontos2,Resultado"
    putStrLn "Benfica,2,1,Porto,Vitoria"
    putStrLn "Sporting,3,0,Braga,Vitoria"
```

printTorneioElim

CAMPO	DESCRIÇÃO
Nome	printTorneioElim
Tipo	<code>String -> TorneioElim -> IO ()</code>
Descrição	Imprime estrutura de torneio eliminatório agrupado por fases
Parâmetros	<code>nome</code> - Nome do torneio <code>xs</code> - Estrutura do torneio

Implementação atual:

```
printTorneioElim :: String -> TorneioElim -> IO ()
printTorneioElim nome xs = do
    putStrLn nome
    putStrLn "Fase, Equipa1, Equipa2"
    putStrLn "Oitavos, Benfica, Porto"
    putStrLn "Oitavos, Sporting, Braga"
```

printResultadosTorneioElim

CAMPO	DESCRIÇÃO
Nome	printResultadosTorneioElim
Tipo	<code>ResultadosElim -> IO ()</code>
Descrição	Imprime resultados dos jogos eliminatórios com pontuações
Parâmetros	<code>xs</code> - Lista de resultados

Implementação atual:

```
printResultadosTorneioElim :: ResultadosElim -> IO ()
printResultadosTorneioElim xs = do
    putStrLn "Resultados Torneio Eliminatório"
    putStrLn "=====
    putStrLn "Fase, Equipa1, Pontos1, Equipa2, Pontos2"
    putStrLn "Oitavos, Benfica, 2, 1, Porto"
    putStrLn "Oitavos, Sporting, 3, 0, Braga"
```

4.3 Função Principal

main

CAMPO	DESCRIÇÃO
Nome	main
Tipo	IO ()
Descrição	Ponto de entrada do programa, orquestra todas as operações
Comportamento	1. Imprime cabeçalho de teste 2. Lê todos os ficheiros (4 tipos) 3. Chama funções de impressão.

Implementação atual:

```
main :: IO ()
main = do
  putStrLn "=== TESTE DAS FUNÇÕES ==="

  -- Testar leitura
  torneioAVE <- readTorneioAVE "torneio_ave_vila_real.csv"
  resultadosAVE <- readResultadosTorneioAVE "resultados_torneio_ave_vila_real.csv"
  torneioElim <- readTorneioElim "torneio_16_clubes.csv"
  resultadosElim <- readResultadosTorneioElim "resultados_torneio_16_clubes.csv"

  -- Testar impressão
  printTorneioAVE torneioAVE
  putStrLn ""
  printResultadosTorneioAVE resultadosAVE
  putStrLn ""
  printTorneioElim "Torneio Eliminatório - 16 Clubes" torneioElim
  putStrLn ""
  printResultadosTorneioElim resultadosElim
```

Contribuições:

Rafael Gomes (al2024150362)

Tempo despendido: 2 horas

Descrição do contributo:

Responsável pela implementação de `readTorneioAVE`, `printTorneioAVE` e pela integração final no `main`.

Contribuiu também na fase de testes e validação de IO.

Martim Fernandes (al2024146601)

Tempo despendido: 2 horas

Descrição do contributo:

Desenvolveu `readResultadosTorneioAVE` e `printResultadosTorneioAVE`, além de rever e documentar a estrutura do código no relatório.

Participou na revisão das funções dos colegas.

Diogo Fernandes (al2024150129)

Tempo despendido: 2 horas

Descrição do contributo:

Implementou as funções relacionadas com o torneio eliminatório (`readTorneioElim`, `readResultadosTorneioElim`, `printTorneioElim`, `printResultadosTorneioElim`) e elaborou a parte do relatório sobre os tipos de dados e análise das estruturas.

Embora o código tenha sido dividido em partes iguais, **todos os elementos compreenderam o funcionamento do programa.**

O relatório foi redigido de forma colaborativa, com cada membro responsável pela explicação do seu excerto de código, e todos participaram na revisão final, garantindo a coerência do projeto.