# ATAD 2017/18

Algoritmos e Tipos Abstratos de Dados

**Enunciado de Mini-Projeto**

P1

## Basket Scores

## 0. Contexto do tema do Projeto

A Federação Portuguesa de Basquetebol ( http://www.fpb) mantem um conjunto de informação detalhada sobre cada jogo (pontos, faltas, assistências e defesas etc..). Esta informação pode ser consultada em:
http://www.fpb.pt/fpb2014/!site.go?s=1&show=my.estatisticas&codigo=Estatisticas.

Os dados são recolhidos em cada jogo, e posteriormente tratados estatisticamente, sendo possível obter indicadores de desempenho por jogo e por jogador.

## 1. Objetivo do Projeto

Pretende-se desenvolver um programa em C para extrair informação útil de ficheiros com dados sobre os jogadores e jogos de basquetebol.

O programa consiste num interpretador de comandos que o utilizador usa para obter diversos tipos de informação, principalmente informação estatística.

### 1.1 Representação dos jogos em Memória

Cada jogador é representado, obrigatoriamente, pela estrutura de dados apresentada na Figura 1, sendo *Date* um tipo de dados apropriado para guardar uma data.

```
typedef struct date{
    int day, month, year;
}Date;

typedef struct player {
    int id;
    char name[50];
    char team[50];
    Date birthDate;
    char gender;
}Player;
```

**Figura 1 – Tipo de dados Player.**

Os resultados que cada jogador obteve em cada jogo são representados, obrigatoriamente, pela estrutura de dados apresentada na Figura 2, sendo o tipo de dados **Statistics**, representado também na Figura 2.

```c
typedef struct statistics {
    int twoPoints; // cesto de 2 pontos
    int threePoints;// cesto de 3 pontos
    int assists;//assistencias
    int fouls;//faltas
    int blocks;//defesas
}Statistics;
typedef struct playerGameStatistics {
    int idPlayer;
    int idGame;
    Statistics statistics;
}PlayerGameStatistic;
```

**Figura 2 – Tipo de dados PlayerGameStatistics.**

É de referir que na implementação deverá garantir-se que todos os dados necessários são guardados em memória, havendo a realocação de memória caso necessário (uso do realloc).

## 1.2 Representação dos jogadores e dos jogos em Ficheiro

Existem dois tipos de ficheiro com dados:

- Ficheiros de dados sobre os Jogadores
- Ficheiros de dados sobre os desempenhos dos jogadores nos jogos.

Ambos os ficheiros se encontram em formato CVS.

Ficheiro dos jogadores (cada linha corresponde a informação sobre um jogador

```
<id_player>; <name>; <team>; <birth_date>; <gender>;
...
```

Ficheiro do desempenho dos jogadores em jogos (cada linha corresponde a informação sobre o desempenho de um jogador num jogo)

```
<id_player>; <id_game>; <two_points>;<three_points>;<assists>;<fouls>;<blocks>;
...
```

O valor <*birth_date*> encontra-se no formato "dd/mm/aaaa".

Pode-se assumir que não existem ficheiros "mal-formados".

### 1.2.1 Ficheiros Disponibilizados

Juntamente com este enunciado são disponibilizados 4 ficheiros exemplo para testes:

- players_1.csv; players_2.csv
- games_1.csv; games_2.csv
- Só poderão no entanto usar-se em pares: (players_1.csv, games_1.csv) e (players_2.csv, games_2.csv)

## 1.4 Comandos

Os comandos que o programa deve implementar estão divididos em quatro grupos:

A. Informação base sobre os Jogadores (só usa informação do ficheiro players)
B. Informação especifica sobre o desempenho dos jogadores durante os jogos (só usa informação do ficheiro games)
C. Informação agregadas sobre jogos e jogadores
D. Indicadores Avançados

Cada comando é representado por uma palavra que pode ser escrita pelo utilizador em maiúsculas ou em minúsculas, não importa.

Quando a memória de Jogadores ou dos Jogos, estiver vazia, a maioria dos comandos limitam-se a escrever imediatamente a mensagem "NAO EXISTE INFORMAÇÂO". Desta forma, as funções que manipulam as estruturas de dados podem assumir que ele não está vazio, o que evita a necessidade de lidar com situações de falhanço.

A forma como os resultados devem ser mostrados no ecrã será descrita na próxima secção.

Os comandos do são os seguintes:

**A. Informação sobre os Jogadores**

- ✓ **LOADP**
  Pede o nome dum ficheiro, abre o ficheiro de Jogadores carrega-o em memória Se o ficheiro não puder ser aberto, escreve "FICHEIRO INACESSIVEL" e estruturas de dados ficam vazios. Se o ficheiro não for do tipo Jogadores escreve "FICHEIRO INCORRETO"
- ✓ **SHOWP**
  - Pergunta se a ordenação é de (A-Z) ou de (Z-A).
    Mostra todos os jogadores ordenados por nome, um por linha, indicando o seu id, em que equipa jogam e a data de nascimento.

- ✓ **TABLE**
  - Mostra da chamada "Table Squads". Trata-se de uma tabela (matrix 4x2) que mostra quantos jogadores existem por escalão etário e por género, tal como se mostra na Figura 3.

| Escalao/ género | Feminino | Masculino |
|---|---|---|
| Sub 14 | 52 | 66 |
| Sub 16 | 65 | 90 |
| Sub 18 | 50 | 81 |
| Seniores | 81 | 45 |

**Figura 3 - Exemplo de uma tabela Table Squads.**

Nota: Deverá usar a seguinte tabela para determinar os escalões:

| Escalão | Data de nascimento |
|---|---|
| Sub 14 | 2004 e 2005 |
| Sub 16 | 2002 e 2003 |
| Sub 18 | 2000 e 2001 |
| Seniores | < 2000 |

- ✓ **SEARCH**
  - Pede o nome de uma equipa e mostra todos os jogadores dessa equipa. Se não existir deve mostrar "EQUIPA INEXISTENTE".

**B. Informação especifica sobre o desempenho dos jogadores durante os jogos**

- ✓ **LOADG**

  Pede o nome dum ficheiro, abre o ficheiro de Jogos carrega-os em memória. Se o ficheiro não puder ser aberto, escreve "FICHEIRO INACESSIVEL" e as estruturas de dados ficam vazias. Se o ficheiro não tiver o formato estabelecido em 1.2 para "games" escreve "FICHEIRO INCORRETO"

- ✓ **SEARCHG**
  - Pede o id do Jogo, caso não existe nenhum registo para o id dado, escreve "JOGO INEXISTENTE", senão: mostra para o jogo identificado:
    - Numero total de pontos marcados por jogo;
    - Numero de jogadores utilizados por jogo;
    - Numero de bloqueios efetuados por jogo;
- ✓ **MVP**
  - Pede o id do Jogo, caso não existe nenhum registo para o id dado, escreve "JOGO INEXISTENTE", senão para o Jogo em questão identifica o melhor jogador em campo e indica o seu índice de MVP. O melhor jogador em campo é aquele com melhor índice de MVP o jogador – (most valuable player) que é calculado pela seguinte formula:
    MVP = 3 x tree_points + 2 x two_points + assists + 2 x blocks – 3 x fouls.

    Nota: Esta formula não corresponde à praticada na federação de basquetebol.

**C. Informação agregada sobre jogos e jogadores**

- ✓ **MFOULP**

  Mostra a média de faltas dadas por jogador por jogo (Exemplo: Tim Smiths deu em media 1.5 faltas por jogo).
- ✓ **MFOULG**

  Mostra a média de faltas cometida por jogo por jogador. (Exemplo: No jogo com id=3 em media cada jogador deu 0.8 faltas)
  Nota: Sugere-se que comece por determinar qual a gama de valores do idGame, de forma a "balizar a sua pesquisa), e em seguida utilize um array auxiliar para guardar o numero de faltas por jogo, que deverá inicializar a zero.
- ✓ **FAIRP**

  Listar as equipas em função do numero médio de faltas por jogo.
  Nota: deverá determinar para cada equipa, quantos faltas ela cometeu, e quantos jogos jogou. E assim determinar a media de faltas por equipa por jogo. Em seguida deverá ordenar a lista crescentemente pelo número medio de faltas.

**D. Indicador Avançados**

**IDEALTEAM**

Pede o escalão da equipa ideal (0-sub14;1- sub16; 2-sub18; 3-senior). Pede o género (F-Feminino, M-Masculino), caso não existam 5 jogadores que preencham os critérios de escalão e género deverá escrever ("NÃO EXISTEM JOGADORES PARA A EQUIPA IDEAL),

senão apresenta a equipa ideal, listando os dados dos 5 jogadores, indicando a função que assumem na equipa.
A formação da equipa ideal deve satisfazer os seguintes critérios:

- 1 jogadores do tipo CENTER – Jogadores que fazem mais assistências por jogo.
- 2 jogadores do tipo SHOOTY GUARDS – Jogadores que fazem mais pontos por jogo.
- 2 jogadores do tipo POINT GUARD" – Jogadores que fazem mais blocos(defesas) por jogo.

  Nota: Em caso de empate, deverão escolher o jogador que participou em mais jogos. Em caso de empate o Jogador mais novo.

## 2 Exemplo de Execução e Formatos de Saída

No ficheiro "results_1.txt" encontra-se um exemplo de execução do programa pretendido e os resultados esperados para os comandos apresentados.

- O exemplo de execução utiliza os ficheiros players_1.csv e games_1.csv;

A informação deve ser apresentada tão fielmente quanto possível ao exemplo.

## 3 Ficheiro base main.c

Com este enunciado é disponibilizado um ficheiro **main.c** contendo a funcionalidade principal do interpretador de comandos. Deverá ser utilizado como a base do desenvolvimento do projeto.

## 4 Relatório

No relatório deverão constar as seguintes secções (para além de capa com identificação dos alunos e índice):

a) Descrição estruturas de dados usadas e respetivas implementações:
b) Para cada comando (exceto LOADP/LOADG/QUIT) fornecer:
   - O código/função respeitante à funcionalidade (excluem-se funções auxiliares, se o seu resultado for óbvio);
   - A complexidade algorítmica da respetiva implementação
c) Limitações:
   - Quais os comandos que apresentam problemas ou não foram implementados;

d) Conclusões:
   - Análise crítica do trabalho desenvolvido.

## 5 Tabela de Cotações e Penalizações

A avaliação do trabalho será feita de acordo com os seguintes princípios:

- Estruturação: o programa deve estar estruturado de uma forma modular e procedimental;
- Correção: o programa deve executar as funcionalidades, tal como pedido.
- Legibilidade e documentação: o código deve ser escrito, formatado e comentado de acordo com o standard de programação definido para a disciplina.

A nota final obtida, cuja tabela de cotações se apresenta a seguir, será ponderada de acordo com os princípios acima descritos.

| **Funcionalidade** | **Cotação (valores)** |
|---|---|
| Leitura de comandos, tratamento de situação de ficheiro inexistente/vazio e saída do programa (QUIT) | 1 |
| Importação de dados (comando LOADP e LOADG) | (1,5+1,5) |
| Ordenação de dados (comando SHOWP) | 1 |
| Pesquisa com contagem (comando TABLE) | 1 |
| Pesquisa (comando SEARCH) | 1 |
| Médias (comandos MFOULP e MFOULG) | 2 |
| Médias e Ordenar (comando FAIRPLAY) | 2 |
| Pesquisa com contagem (comando SEARCHG) | 1,5 |
| Cálculos agregados e Máximo (comando MVP) | 1,5 |
| Pesquisa, ordenação, media, cálculos agregados IDEAL TEAM | 3 |
| Relatório | 3 |
| **TOTAL** | **20** |

A seguinte tabela contém penalizações a aplicar:

| **Descrição** | **Penalização (valores)** |
|---|---|
| Uso de variáveis globais | até 2 |
| Não separação de funcionalidades em funções | até 3 |
| Não utilização de Módulos | até 2 |

## 6 Instruções e Regras Finais

O IDE a utilizar fica ao critério dos alunos, mas, caso não utilizem o IDE usado na disciplina (i.e., Visual Studio), terão que, **antes de submeter, criar os respetivos projetos finais no IDE** Visual Studio 2017.

O não cumprimento das regras a seguir descritas implica uma penalização na nota do trabalho prático. Se ocorrer alguma situação não prevista nas regras a seguir expostas, essa ocorrência deverá ser comunicada ao respetivo docente de laboratório de ATAD.

### Regras:

a) O Mini-Projeto deverá ser elaborado por **dois alunos do mesmo docente de laboratório**.

b) A nota do Mini-Projeto será atribuída individualmente a cada um dos elementos do grupo após a discussão. As discussões poderão ser orais e/ou com perguntas escritas. As orais poderão ser feitas com todos os elementos do grupo presentes em simultâneo ou individualmente.

c) A apresentação de relatórios ou implementações plagiadas leva à imediata atribuição de nota zero a todos os trabalhos com semelhanças, quer tenham sido o original ou a cópia.

d) No rosto do relatório e nos ficheiros de implementação deverá constar o número, nome e turma dos autores e o nome do docente a que se destina.

e) O trabalho deverá ser submetido no moodle, no link do respetivo docente de laboratórios criado para o efeito, até às **11:00 do dia 30 de Abril**. Para tal terão que criar uma pasta com o nome: **nomeAluno1_númeroAluno1-nomeAluno2_númeroAluno2**, onde colocarão o ficheiro do relatório em formato **pdf** e uma pasta com o projeto Visual Studio da implementação das aplicações a desenvolver. Os alunos terão de submeter essa **pasta compactada em formato ZIP**. Apenas será permitido submeter um ficheiro.

f) Não serão aceites trabalhos entregues que não cumpram na íntegra o ponto anterior.

g) As datas das discussões serão publicadas após a entrega dos trabalhos.

(fim de enunciado)