Instituto Federal do Espírito Santo — IFES (Campus Serra)
Coordenadoria de Informática — Cin

Trabalho de Programação

Campeonato Computacional de Futebol – Parte I

Valor: 15 pontos
Deadline: 07 de novembro de 2025

Prof. Thiago M. Paixão
thiago.paixao@ifes.edu.br

1 Objetivo

O trabalho prático de programação consiste em implementar um sistema simplificado em linguagem C de gerenciamento de dados de um campeonato "computacional" de futebol. Os dados são armazenados (persistidos) em um arquivo CSV e carregados em uma estrutura de dados específica (memória) para execução do sistema. O campeonato é composto por 10 clubes (indexados de 0 a 9): JAVAlis, ESCorpiões, SemCTRL, GOrilas, PYthons, SeQueLas, NETunos, LOOPardos, RUSTicos, REACTivos.

A competição é disputada em um turno de pontos corridos, ou seja, todos jogam contra todos (sem repetição de jogos) e o campeão, obviamente, é o time com mais pontos. A classificação dos times é definida pelos seguintes critérios:

- **V**: vitórias;
- **E**: empates;
- **D**: derrotas;
- **GM**: gols marcados;
- **GS**: gols sofridos;
- **S**: saldo de gols ($GM - GS$);
- **PG**: pontos ganhos ($3V + E$);

Para esta entrega específica (Parte I), o foco é a consulta de partidas e resultados, e a obtenção da tabela de pontuação das equipes envolvidas. A estrutura de dados será composta por um vetor estático de registros (`struct`) de partidas, com um tamanho pré-definido que atenda às necessidades esperadas para o volume de dados a ser gerenciado. A escolha pelo vetor estático é baseada na simplicidade da implementação e na previsibilidade do consumo de memória.

As principais competências a serem desenvolvidas neste trabalho incluem:

- Alocação dinâmica.
- Modularização e TADs.
- Manipulação de arquivos.

2 O Sistema

2.1 Comportamento básico

O sistema deve permitir o registro de partidas e resultados, e a consulta da tabela de classificação.

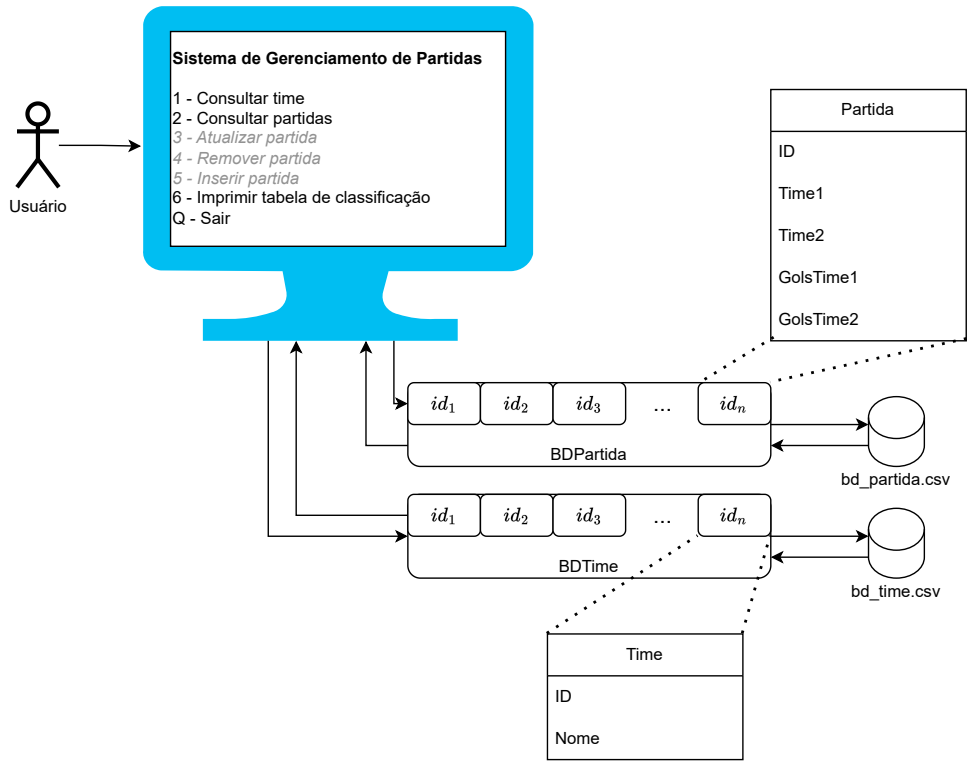

Figura 1: Sistema de gerenciamento de registro de partidas de futebol.

O comportamento básico do sistema é ilustrado na Figura 1. Ao rodar o sistema, o usuário acessa um menu com um total de seis opções, conforme indicado na figura, sendo três delas desabilitadas nesta primeira etapa do trabalho. Deste modo, devem ser implementadas nesta entrega as funcionalidades de consultar time, consultar partidas e imprimir a tabela de classificação (opções `1`, `2` e `6`, detalhadas na Seção 2.3), bem como a ação de encerrar a execução do sistema (`Q`).

Para este trabalho, simularemos um banco de dados composto por duas tabelas que persistem os dados em arquivos CSV[1]. Ao iniciar o sistema, o conteúdo dos arquivos `times.csv` e `partidas.csv` será carregado para as estruturas de dados em memória responsáveis por gerenciar as informações de times e partidas, respectivamente. Essas estruturas funcionarão como um espelho dos arquivos CSV até que alguma operação de inserção, atualização ou exclusão seja realizada pelo usuário, o que será implementado na próxima fase do trabalho.

2.2 Banco de dados

2.2.1 Times e partidas

Arquivo times.csv

Este arquivo contém os dados referentes aos times de futebol cadastrados no sistema. Cada linha representa um time com os seguintes campos:

- ID: Identificador único de cada time garantindo que não existam duplicatas.
- Nome: Nome completo do time de futebol.

Arquivo partidas.csv

Este arquivo armazena os registros das partidas realizadas entre os times cadastrados. Cada linha representa uma partida com os seguintes campos:

[1]Todos os arquivos necessários estão disponíveis em NESTE LINK.

- ID: Identificador único da partida.
- Time1: O ID do primeiro time participante.
- Time2: O ID do segundo time participante.
- GolsTime1: Quantidade de gols marcados pelo primeiro time.
- GolsTime2: Quantidade de gols marcados pelo segundo time.

Dessa forma, o banco de dados simula a relação entre as tabelas de times e partidas, permitindo que o sistema realize consultas e operações sobre as informações armazenadas de forma persistente.

```
ID,Time1,Time2,GolsTime1,GolsTime2
0,0,1,10,0
1,0,2,3,2
2,0,3,3,4
```

Figura 2: Exemplo de banco de dados em arquivo (CSV) com dados de partidas.

2.2.2 Conjuntos de dados para teste

Para garantir a robustez e corretude do sistema, serão disponibilizado três arquivos de partidas distintos para testes. Cada conjunto representa um cenário específico da competição, permitindo validar o comportamento do programa em diferentes estágios. O arquivo `times.csv` permanece o mesmo em todos os cenários, enquanto o arquivo `partidas.csv` deve ser substituído por uma das seguintes versões para simular cada caso:

- **Cenário 1: Campeonato Vazio (`partidas_vazio.csv`)**
  Este arquivo está vazio, contendo apenas o cabeçalho. Seu propósito é testar o comportamento inicial do sistema, onde nenhum jogo foi disputado. Espera-se que funcionalidades como "Imprimir tabela de classificação" exibam todos os times com suas estatísticas (pontos, vitórias, gols, etc.) devidamente zeradas.
- **Cenário 2: Campeonato em Andamento (`partidas_parcial.csv`)**
  Contém um subconjunto das partidas totais da competição. Este cenário simula o campeonato durante seu andamento e serve como caso de teste para validar se os cálculos de pontos, saldo de gols e demais estatísticas estão sendo acumulados corretamente após cada partida processada.
- **Cenário 3: Campeonato Finalizado (`partidas_completo.csv`)**
  Inclui o registro de todas as partidas planejadas. Este conjunto de dados permite verificar os resultados finais da competição e validar a totalização das estatísticas de todos os times. Também servirá como base para futuras funcionalidades, como a ordenação da tabela de classificação final.

2.2.3 TAD `Time`

O TAD `Time` é uma abstração que modela uma única equipe de futebol, encapsulando seus dados de identificação e seu desempenho na competição. Ele armazena os dados de identificação lidas do arquivo `times.csv` – um identificador único (ID) e um nome – e, de forma complementar, acumula as estatísticas de desempenho do time calculadas a partir dos resultados das partidas: vitórias (V), empates (E), derrotas (D), gols marcados (GM) e gols sofridos (GS).

A interface pública do TAD, além de fornecer acesso a esses dados, também oferece funções para obter dados derivados, como os pontos ganhos (PG) e o saldo de gols (SG), que são calculados sob demanda

para garantir a consistência e a integridade das informações em todos os momentos. O TAD `Time` ↪ funciona como a estrutura de dados elementar que é criada, armazenada e manipulada pelo TAD `BDTimes` (Seção 2.2.4).

2.2.4 TAD `BDTimes`

O TAD `BDTimes` atua como um gerenciador para a coleção de todos os times do sistema. Sua principal responsabilidade é carregar os dados iniciais a partir do arquivo `times.csv`, criando uma instância do TAD `Time` para cada registro e armazenando-as em uma estrutura de dados interna. Toda a manipulação dessa coleção – como busca, listagem ou atualização das estatísticas de um time específico – é realizado por meio de uma interface pública, que abstrai os detalhes de implementação.

2.2.5 TAD `Partida`

O TAD `Partida` é uma abstração para representar as informações de um único jogo de futebol. Ele encapsula todos os campos de um registro do arquivo `partidas.csv`: o identificador da partida (`ID`), os identificadores dos times envolvidos (`Time1` e `Time2`), e a quantidade de gols marcada por cada um dos times (`GolsTime1` e `GolsTime2`). Este TAD serve como a estrutura de dados fundamental que será gerenciada pelo TAD `BDPartidas` (Seção 2.2.6).

2.2.6 TAD `BDPartidas`

O TAD `BDPartidas`, por sua vez, gerencia a coleção de todas as partidas carregadas do arquivo `partidas.csv`. Sua interface deve oferecer um conjunto de operações para interagir com essa coleção, incluindo funções para busca e listagem de jogos. A principal responsabilidade do TAD é fornecer ao sistema um acesso estruturado aos resultados dos jogos, permitindo que outras partes do sistema, como o `BDTimes` (Seção 2.2.4), processem esses dados para calcular as estatística de desempenho das equipes.

2.3 Funcionalidades

A seguir, são descritas as funcionalidades obrigatórias do sistema. As descrições são fornecidas de forma geral, cabendo ao estudante decidir pela melhor forma de implementação considerando eficiência e usabilidade.

Consultar time A funcionalidade de consulta de time permite ao usuário verificar o desempenho detalhado de uma ou mais equipes na competição. A busca é realizada a partir do nome do time, suportando a pesquisa por prefixo. Para cada time encontrado que corresponda ao prefixo informado, o sistema exibirá um resumo completo de suas estatísticas, incluindo o número de vitórias (V), empates (E), derrotas (D), gols marcados (GM), gols sofridos (GS), o saldo de gols (S) e os pontos ganhos (PG). Caso nenhum time seja encontrado com o nome ou prefixo fornecido, uma mensagem informativa de erro será exibida ao usuário. A Figura 3 ilustra o fluxo de execução desta funcionalidade.

Consultar partidas A funcionalidade de consulta de partidas permite buscar os resultados das partidas utilizando o nome do time mandante, time visitante ou ambos. Baseado na informação fornecida pelo usuário, o sistema deve imprimir todos os registros correspondentes. Caso nenhuma partida seja encontrada, o sistema deve informar o usuário com uma mensagem de erro. A busca será baseada em prefixo, ou seja, se o usuário fornecer apenas uma parte do nome do time, o sistema deve retornar todos os registros que correspondam ao prefixo informado. A Figura 4 ilustra um possível fluxo de execução desta funcionalidade.

```
[Sistema]
Digite o nome ou prefixo do time:

[Usuário]
Se

[Sistema]
ID  Time            V   E   D   GM  GS  S   PG
2   SemCTRL         4   1   5   12  20  -8  13
5   SeQueLas        8   0   2   27  5   22  24
...
```

Figura 3: Simulação da execução de consulta de desempenho de time.

```
[Sistema]
Escolha o modo de consulta:
1 - Por time mandante
2 - Por time visitante
3 - Por time mandante ou visitante
4 - Retornar ao menu principal

[Usuário]
3

[Sistema]
Digite o nome:

[Usuário]
NET

[Sistema]
ID  Time1              Time2
23  NETunos    1 x 1   SemCTRL
47  REACTivos  4 x 2   NETunos
...
```

Figura 4: Simulação da execução de consulta de placares.

Imprimir tabela de classificação A funcionalidade de impressão da tabela de classificação permite exibir todos os times cadastrados, mostrando o desempenho acumulado com base nas partidas realizadas, nessa primeira parte não é necessário ordenar a classificação. Para cada time, o sistema deve apresentar os seguintes campos: ID, Time, V (vitórias), E (empates), D (derrotas), GM (gols marcados), GS (gols sofridos), S (saldo de gols) e PG (pontos ganhos). O sistema deve organizar a impressão seguindo a ordem dos IDs dos times e, caso haja um grande número de registros, é recomendado que a saída seja paginada para não sobrecarregar a interface. A Figura 5 ilustra a saída do sistema quando essa funcionalidade é solicitada.

```
[Sistema]
Imprimindo classificação...

ID  Time            V   E   D   GM  GS  S   PG
0   JAVAlis         10  0   0   45  0   45  30
1   ESCorpiões      5   0   5   21  22  -1  15
2   SemCTRL         7   2   1   33  6   27  23
...
```

Figura 5: Simulação da execução de impressão de lista de times e suas pontuações.

3 Requisitos de implementação

Neste trabalho, você terá a flexibilidade de implementar/adequar os módulos e os Tipos Abstratos de Dados (TADs) que considerar necessários. No entanto, é fundamental que o programa principal esteja implementado no arquivo `main.c`. Além disso, a estrutura do código deve ser modular. É importante que cada módulo tenha uma responsabilidade clara e que a comunicação entre os diferentes componentes do programa ocorra de forma eficiente.

Note que, apesar das sugestões de implementação (fluxo de execução) em alto nível, os detalhes de implementação devem ser decididos por vocês. Você é encorajado a adicionar e/ou modificar funcionalidades de modo a melhorar a experiência do usuário ou otimizar a execução. Não se deve, contudo, reduzir a quantidade de funcionalidades já previstas, nem reduzir o escopo do projeto.

4 Critérios de avaliação

A avaliação deste trabalho levará em consideração os seguintes critérios que juntos somam **15 pontos**:

1. Funcionalidades: Até **9 pontos** serão atribuídos à implementação adequada das 3 funcionalidades requeridas (3 pontos por funcionalidade).
2. Lógica e organização do código: Até **2 pontos** serão concedidos pela clareza, organização e boas práticas de codificação no projeto. Isso inclui a estruturação adequada do código (modularização), nomes significativos para variáveis e funções, e formatação consistente.
3. Documentação do `README.md`: Até **2 pontos** serão atribuídos à qualidade do arquivo `README.md` presente no repositório. Este arquivo deve ser descritivo e informativo, contendo instruções claras sobre como executar e utilizar o projeto. Deve incluir informações detalhadas sobre a estrutura do repositório, apresentar os principais TADs utilizados e listar as principais decisões de implementação tomadas ao longo do desenvolvimento.
4. Documentação interna do código: Até **1 ponto** será atribuído à qualidade da documentação incorporada diretamente ao código. Essa documentação deve ser composta por comentários significativos que expliquem a lógica por trás das implementações, facilitando a compreensão do funcionamento do projeto e promovendo a manutenção do código.
5. Robustez: A nota de robustez ($R \in [0, 1]$) será atribuída com base na presença de falhas críticas (por exemplo, falha de segmentação) ou não críticas (por exemplo, *memory leakage*) no sistema.
6. Dias de atraso: Serão contabilizados os dias de atraso (D) para efeito de desconto na nota total do trabalho.

A nota final do trabalho será calculada pela equação:

$$\text{nota} = \left(1 - \frac{2^D - 1}{31}\right) \times R \times P, \tag{1}$$

onde P é a soma dos pontos dos critérios 1 a 4. É importante ressaltar que a nota do trabalho será zerada caso o atraso ultrapasse 5 dias.

Importante: O programa será testado num ambiente Linux Ubuntu 22.04 com GCC 11. Recomendo FORTEMENTE desenvolver e testar nesse ambiente.

5 O que entregar?

1. Um link para um repositório .git com o arquivo `bd_partidas.csv`, `bd_classificacao.csv` e código-fonte do projeto: `Makefile` e arquivos `.c` e `.h`.
2. A documentação/relatório será feita no arquivo `README.md` do repositório e deverá explicar o passo-a-passo para executar o programa, os principais TADs e as principais decisões de implementação.

Bom trabalho!