Trabalho Prático 2

Implementação de um sistema de gestão da Fórmula 1

1. Informação geral

O **Trabalho Prático 2** aplica conceitos de **Programação Orientada a Objetos** e requer a implementação de classes baseadas em **listas** e **filas**, além dos **vetores**.

Este trabalho deve ser desenvolvido de forma **autónoma**, por cada grupo, e entregue até a data limite estabelecida.

- É permitida a consulta de informação em diversas fontes, mas o **código submetido deve ser exclusivamente da autoria do grupo**.
- Cópias detetadas serão penalizadas.
- Todos os elementos do grupo devem ser capazes de **explicar e modificar o código submetido**, caso contrário, haverá penalização.
- A submissão deve ser feita via **Moodle**, até **30 de março, às 23:59h**.

2. Conceito

Os proprietários da **Fórmula 1** pretendem obter um sistema para gerir os pilotos, as equipas (construtores), circuitos e as corridas.

3. Implementação do trabalho

O arquivo comprimido **`ESDA_2025_T2.zip`** contém os ficheiros necessários para a realização deste trabalho, nomeadamente:

- `F1.hpp`: definição das classes para representação do sistema (`Drive`, `Constructor`, `DriverManagement`, `ConstructorManagement`, `Circuit`, `CircuitManagement`, `Race` e `RaceManagement`) e as estruturas `DriCons` e `DriResult`.
- `F1.cpp`: implementação dos métodos relativos às classes definidas em `F1.hpp`.
- `F1_test.cpp`: programa principal para testes das funções implementadas.
- `alldrivers.csv`: ficheiro de texto com a informação de todos os pilotos.
- `allconstructors.csv`: ficheiro de texto com a informação de todos os construtores.
- `numbers.csv`: ficheiro de texto com a informação de todos números utilizados pelos pilotos.
- `allConstruDrivers.csv`: ficheiro de texto com a informação da associação de pilotos e construtores por ano.
- `allCircuits.csv`: ficheiro de texto com a informação de todos os circuitos.
- `allraces.csv`: ficheiro de texto com a informação de todas as corridas (de 2004 a 2009).
- `allresults.csv`: ficheiro de texto com a informação de todos os resultados das corridas (de 2004 a 2009).

Notas importantes:

1. **Apenas deverá ser alterado o ficheiro `F1.cpp`, sendo somente necessário incluir a implementação de cada função na submissão do código em CodeRunner, no Moodle.**
2. **Cada atributo e método das classes definidas apresenta detalhes adicionais junto a cada um deles em `F1.hpp`.**

O ficheiro `F1.hpp` contém as classes `Driver`, `Constructor`, `DriverManagement`, `ConstructorManagement`, `Circuit`, `CircuitManagement`, `Race` e `RaceManagement`, e a estrutura `DriCons` e `DriResult`. A primeira permite caracterizar cada o piloto, a segunda caracteriza cada construtor, a terceira permite a gestão de todos os pilotos, a quarta permite a gestão de todos os construtores, a quinta permite caracterizar cada circuito, a sexta faz a gestão dos circuitos em lista, a sétima caracteriza uma corrida, e a última permite a gestão em lista das corridas. As estruturas: a primeira associa um intervalo de tempo a um piloto e a segunda contém a informação do resultado de um piloto para uma determinada corrida.

Classe `Driver`

Os objetos da classe `Driver` têm os seguintes atributos:

3. Identificador único do piloto (`driverId`)
4. Código de 3 caracteres do piloto (`code`)
5. Nome do piloto (`name`)
6. Vetor de inteiros com os números já utilizados pelo piloto (`numbers`)
7. Data de nascimento do piloto (`dateOfBirth`)
8. Nacionalidade do piloto (`nationality`)

Classe `Constructor`

Os objetos da classe `Constructor` têm os seguintes atributos:

1) Identificador único do construtor (`constructorId`)
2) Nome do construtor (`name`)
3) Nacionalidade do construtor (`nationality`)
4) Vetor de apontadores para a estrutura `DriCons`, pilotos que já representaram o construtor (`Drivers`)

Classe `DriverManagement`

Os objetos da classe `DriverManagement` possuem um vetor de apontadores para objetos da classe `Driver`, representando todos os pilotos.

Classe `ConstructorManagement`

Os objetos da classe `ConstructorManagement` possuem um vetor de apontadores para objetos da classe `Driver`, representando todos os construtores.

Classe `Circuit`

Os objetos da classe `Circuit` têm os seguintes atributos:

1) Identificador único do circuito (`circuitId`)
2) Nome do circuito (`name`)
3) Local do circuito (`location`)
4) País do circuito (`country`)
5) Altitude em que se encontra o circuito (`alt`)

Classe `CircuitManagement`

Os objetos da classe `CircuitManagement` possuem uma lista de apontadores para objetos da classe `Circuit`, representando todos os circuitos.

Classe `Race`

Os objetos da classe `Race` têm os seguintes atributos:

1) Identificador único da corrida (`raceId`)
2) Ano (`season`)
3) Etapa (`round`)
4) Apontador para o circuito (`circuit`)
5) Data (`date`)
6) Lista com os resultados da corrida (`listRaceResults`)

Classe `RaceManagement`

Os objetos da classe `RaceManagement` possuem uma lista de apontadores para objetos da classe `Race`, representando todos os resultados.

As funções a implementar neste trabalho correspondem a métodos definidos em cada classe.

Classe `RaceManagement`

1. `int` **`numberRacePerSeason`**`(int nRaces);`
   Número de épocas que tiveram um determinado número de corridas. Retorna o número de épocas ou -1 no caso de `nRaces` *não ser válido.*

2. `queue<string>` **`seasonGrandPrix`**`(int year);`
   Apresenta o nome de todos os grandes prémios de uma determinada época numa fila, Retorna a fila com os nomes dos grandes prémios, ordenados pelo número da corrida, de modo que a primeira corrida do ano seja a primeira a sair da fila. Retorna -1 se o ano for inválido.

3. `int` **`uploadFromFile`**`(string filename, DriverManagement &driManager, ConstructorManagement &consManager);`
   Atualiza os resultados dos pilotos nas corridas, lendo o conteúdo do ficheiro de texto `filename`*. Retorna o número de resultados inseridos ou -1 em caso de erro. Cada linha do ficheiro contém a informação necessária no seguinte formato:* *`raceId;driverId;constructorId;number;grid;position;laps;statusId`*.

4. `string` **`driverStatus`**`(string status, int &num);`
   Determina qual o piloto que teve a maior frequência de um determinado tipo de ocorrência em corridas (como acidente, pneu furado, etc. No caso de a contagem resultar em empate, é selecionada a que se encontra alfabeticamente à frente. Retorna o nome do piloto e, pelo parâmetro de entrada `num`*, o número de ocorrências. Retorna -1 se o tipo de ocorrência for inválido.*

5. `int` **`scoreCareer`**`(int driveI, DriverManagement &drivManager, int &ngrids);`
   Determina o total de pontos e pole position de um piloto. Retorna o número de pontos ou -1 caso o piloto (`driverId`*) não seja válido, e o numero de poles position pelo parâmetro de entrada* `ngrids`*.*

   Tenha em atenção que o sistema de pontuação mudou ao longo dos anos:

1950-1959	***1960-1990***	***1991-2002***
1º – 8 pontos	*1º – 9 pontos*	*1º – 10 pontos*
2º – 6 pontos	*2º – 6 pontos*	*2º – 6 pontos*
3º – 4 pontos	*3º – 4 pontos*	*3º – 4 pontos*
4º – 3 pontos	*4º – 3 pontos*	*4º – 3 pontos*
5º – 2 pontos	*5º – 2 pontos*	*5º – 2 pontos*
	6º – 1 ponto	*6º – 1 ponto*
2003-2009	***2010-2024***	
1º – 10 pontos	*1º – 25 pontos*	
2º – 8 pontos	*2º – 18 pontos*	
3º – 6 pontos	*3º – 15 pontos*	
4º – 5 pontos	*4º – 12 pontos*	
5º – 4 pontos	*5º – 10 pontos*	
6º – 3 pontos	*6º – 8 pontos*	
7º – 2 pontos	*7º – 6 pontos*	
8º – 1 ponto	*8º – 4 pontos*	
	9º – 2 pontos	
	10º – 1 ponto	

Nota: Os ficheiros de entrada e casos de teste em que serão avaliadas as funções submetidas poderão apresentar conteúdo diferente e incluir casos limite (por exemplo, argumentos de funções com gamas não previstas). Como tal, é sua responsabilidade garantir que os argumentos são devidamente testados de forma a aceitá-los apenas quando válidos.

4. Teste da biblioteca de funções

A biblioteca pode ser testada executando o programa `F1_test`. Existe um teste por cada função a implementar e que determina se essa função tem o comportamento esperado. Note que os

testes não são exaustivos. Por isso, os testes devem ser considerados apenas como um indicador de uma aparente correta implementação das funcionalidades esperadas.

Se as funções passarem nos testes unitários incluídos, o programa `F1_test`, quando executado, deverá apresentar o seguinte resultado:

```
INICIO DOS TESTES


Importou: 861 drivers
Importou: 212 constructors
Importou: 77 cuircuits
Importou: 107 races

...verifica_numberRacePerSeason ( 18 races ):  Total de épocas, retorno =3 (ok)
...verifica_numberRacePerSeason ( 2 races ):  Total de épocas, retorno =0 (ok)
OK: verifica_numberRacePerSeason passou


...verifica_seasonGrandPrix(época 2004): Numero de Grande prémios (=18) (ok)
...verifica_seasonGrandPrix(época 2020): Grande prémios (=Albert Park Grand Prix Circuit Sepang International Circuit Bahrain International Circuit Autodromo Enzo e Dino Ferrari Circuit de Barcelona-Catalunya Circuit de Monaco N□rburgring Circuit Gilles Villeneuve Indianapolis Motor Speedway Circuit de Nevers Magny-Cours Silverstone Circuit Hockenheimring Hungaroring Circuit de Spa-Francorchamps Autodromo Nazionale di Monza Shanghai International Circuit Suzuka Circuit Aut□dromo Jos□ Carlos Pace) (ok)
Albert Park Grand Prix Circuit Sepang International Circuit Bahrain International Circuit Autodromo Enzo e Dino
Ferrari Circuit de Barcelona-Catalunya Circuit de Monaco N□rburgring Circuit Gilles Villeneuve Indianapolis Motor Speedway Circuit de Nevers Magny-Cours Silverstone Circuit Hockenheimring Hungaroring Circuit de Spa-Francorchamps Autodromo Nazionale di Monza Shanghai International Circuit Suzuka Circuit Aut□dromo Jos□ Carlos Pace
OK: verifica_seasonGrandPrix passou


...verifica_uploadFromFile:  retorno (=2213) (ok)
...verifica_uploadFromFile ( grande premio de Spain de 2009):  Numero de pilotos que participaram na corrida (=20) (ok)
...verifica_uploadFromFile ( grande premio de Spain de 2009):    1º classificado (=Rubens Barrichello) (ok)
OK: verifica_uploadFromFile passou


...verifica_driverStatus: Quem teve mais problemas com o motor  (piloto =Kimi Räikkönen) (ok)
...verifica_driverStatus:  O numero de problemas de motor (=7) do Kimi Räikkönen é o esperado (ok)
OK: verifica_driverStatus passou


...verifica_scoreCareer(Kimi Räikkönen): Pontuação total da carreira (=455) (ok)
...verifica_scoreCareer(Kimi Räikkönen): Pole position (=14) (ok)
OK: verifica_scoreCareer passou


FIM DOS TESTES: Todos os testes passaram
```

5. Ferramenta de desenvolvimento

A utilização de um IDE ou do Visual Studio Code é aconselhável no desenvolvimento deste trabalho, uma vez que permite fazer depuração de uma forma mais eficaz. Poderá encontrar informações sobre a utilização do Visual Studio Code num breve tutorial disponibilizado no Moodle.

É possível implementar as funções solicitadas diretamente no CodeRunner, sendo aconselhável consultar os ficheiros fornecidos, de modo a compreender todo o contexto do trabalho a ser realizado.

6. Avaliação

A classificação do trabalho será baseada em:

1. **Implementação**: avaliada automaticamente no **Moodle**.
    - Se não compilar, a nota será **0**.
    - Testes adicionais serão aplicados.
2. **Avaliação Oral**: explicação e capacidade de modificar o código.
    - **100%**: Domina totalmente o código
    - **75%**: Algumas falhas
    - **50%**: mais ao menos
    - **25%**: Muitas falhas
    - **0%**: Graves lacunas

A nota final (**MTP2**) é dada por:

$$MTP2 = Implementação \times Avaliação\ oral$$

7. Submissão da resolução

A submissão é apenas possível através do Moodle e até à data indicada no início do documento. A submissão da implementação das funções deverá ser realizada através do CodeRunner, nos espaços preparados no Moodle. Só é necessário um elemento do grupo inserir a solução.