Trabalho Prático 1

Implementação de um sistema de gestão da Fórmula 1

1. Informação geral

O **Trabalho Prático 1** aplica conceitos de **Programação Orientada a Objetos** e requer a implementação de classes baseadas em **vetores**, uma estrutura de dados linear.

Este trabalho deve ser desenvolvido de forma **autónoma**, por cada grupo, e entregue até à 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 **16 de março, às 23:59h**.

2. Conceito

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

3. Implementação do trabalho

O arquivo comprimido **`ESDA_2025_T1.zip`** contém os ficheiros necessários para a realização deste trabalho, nomeadamente:

- `F1.hpp`: definição das classes para representação do sistema (`Drive`, `Constructor`, `DriverManagement` e `ConstructorManagement`) e a estrutura `DriCons`.
- `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.

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` e `ConstructorManagement`, e a estrutura `DriCons`. A primeira permite caracterizar cada piloto, a segunda caracteriza cada construtor, a terceira permite a gestão de todos os pilotos e a última permite a gestão de todos os construtores; a estrutura associa um intervalo de tempo a um piloto.

Classe `Driver`

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

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

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

Classe `Driver`

1. `int` **`addNumber`**`();`
   Adiciona um novo número ao vetor de números. O número deve estar compreendido entre 1 e 199. Retorna zero em caso de sucesso, -1 em caso de erro ou 1 caso o número já exista.

2. `void` **`displayDriverInfo`**`(ostream& o) const;`
   Apresenta a informação do piloto. Inclui imprimir: o identificador, sigla de 3 caracteres, nome, números utilizados, data de nascimento e nacionalidade.

Classe `Constructor`

3. `int` **`addDriver`**`(int driveId, int year, DriverManagement &driveManager);`
   Adiciona um novo piloto ao Construtor ou adiciona um novo ano a um piloto que já existe no Construtor. Retorna 0 se os novos pilotos ou um novo ano forem adicionados com sucesso, 1 se os novos pilotos ou o novo ano já existirem, ou -1 se os parâmetros são inválidos ou o piloto não for encontrado.

Classe `ConstructorManagement`

4. `int` **`updateConstructorDrivers`**`(string filename, DriverManagement& manager);`
   Atualiza os pilotos nos construtores, lendo o conteúdo do ficheiro de texto `filename`*. Retorna 0 se a atualização teve sucesso, ou -1 em caso de erro.*
   Cada linha do ficheiro contém a informação necessária no seguinte formato: `constructorId;driverId;year`*.*

5. `vector<string>` **`DriversAndConstructorsOfOneYear`**`(int year, vector<string> &vConstr);`
   Cria duas listas, uma com todos os pilotos de um determinado ano e outra com todos os construtores desse ano. As listas têm de estar ordenadas por ordem alfabética. A lista de pilotos é o vetor de retorno e a lista de construtores é feita no parâmetro de entrada `vConstr`*.*

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

...verifica_addNumber:  Total dos numeros do carro do Max Verstappen, retorno =2 (ok)
...verifica_addNumber: Os numeros do carro do Max Verstappen, retorno =33 1 (ok)
OK: verifica_addNumber passou


...verifica_displayDriverInfo: Informação do Lewis Hamilton está correta (ok)
Displaying driver info:
-----DriverId: 1
-----Code: HAM
-----Name: Lewis Hamilton
-----used numbers: 22 2 1 3 4 10 44
-----Date of birth: 1/7/1985
-----Country: British
OK: verifica_displayDriverInfo passou

...verifica_addDriver(Inserir Felipe Massa na Ferrari em 2008): Retorno =0 (ok)
...verifica_addDriver(Inserir Felipe Massa na Ferrari em 2008): O numero de pilotos na Ferrari=1
(ok)
...verifica_addDriver(Inserir Felipe Massa na Ferrari em 2008):  Inseriu com sucesso
...verifica_addDriver(Inserir Felipe Massa na Ferrari em 2009): Retorno =0 (ok)
...verifica_addDriver(Inserir Felipe Massa na Ferrari em 2009): O numero de pilotos na Ferrari=1
(ok)
...verifica_addDriver(Inserir Felipe Massa na Ferrari em 2009): Inseriu com sucesso
...verifica_addDriver(Inserir Fernando Alonso na Ferrari em 2009): Retorno =0 (ok)
...verifica_addDriver(Inserir Fernando Alonso na Ferrari em 2009): O numero de pilotos na Ferrari=2
(ok)
...verifica_addDriver(Inserir Fernando Alonso na Ferrari em 2009): Inseriu com sucesso
OK: verifica_addDriver passou

...verifica_updateConstructorDrivers:  retorno =0 (ok)
...verifica_updateConstructorDrivers(Construtor Life): O numero de pilotos na Life=2 (ok)
...verifica_addDriver(Construtor Life(Gary Brabham and Bruno Giacomelli in 1990)): Inseriu com
sucesso
OK: verifica_updateConstructorDrivers passou

...verifica_DriversAndConstructorsOfOneYear(2018): Numero de pilotos=20 (ok)
...verifica_DriversAndConstructorsOfOneYear(2018):  Nomes dos pilotos(=Brendon Hartley-Carlos
Sainz-Charles Leclerc-Daniel Ricciardo-Esteban Ocon-Fernando Alonso-Kevin Magnussen-Kimi
Räikkönen-Lance Stroll-Lewis Hamilton-Marcus Ericsson-Max Verstappen-Nico Hülkenberg-Pierre Gasly-
Romain Grosjean-Sebastian Vettel-Sergey Sirotkin-Sergio
Pérez-Stoffel Vandoorne-Valtteri Bottas) (ok)
...verifica_DriversAndConstructorsOfOneYear(2018): Numero de Construtores=10 (ok)
...verifica_DriversAndConstructorsOfOneYear(2018): Nomes dos Construtores(=Ferrari-Force India-
Haas F1 Team-McLaren-Mercedes-Red Bull-Renault-Sauber-Toro Rosso-Williams) (ok)
OK: verifica_DriversAndConstructorsOfOneYear 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
    - **40%**: Muitas falhas
    - **0%**: Graves lacunas

A nota final (**MTP1**) é dada por:

$$\text{MTP1} = \text{Implementação} \times \text{Avaliação oral}$$

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.