Trabalho Prático 5

Implementação de uma aplicação de gestão de séries televisivas

1. Informação geral

O Trabalho Prático 5 aplica conceitos de Programação Orientada a Objetos e consiste na implementação de classes baseadas em diferentes estruturas de dados, que serão escolhidas pelos estudantes de modo a obter a melhor eficiência possível.

Este trabalho deverá ser feito de forma autónoma, por cada grupo, até à data-limite estabelecida. A consulta de informação nas diversas fontes disponíveis é aceitável. No entanto, o código submetido deverá ser apenas da autoria dos elementos do grupo e quaisquer cópias detetadas serão devidamente penalizadas. A incapacidade de explicar o código submetido por parte de algum elemento do grupo implicará também uma penalização.

O prazo limite para submissão (através do Moodle) é o dia **19 de maio às 23:59h**.

2. Conceito

Uma plataforma de gestão de séries televisivas pretende obter uma aplicação que permita gerir as séries.

3. Implementação do trabalho

O arquivo comprimido **`EDA_2024_TP5.zip`** contém os ficheiros necessários para a realização deste trabalho, nomeadamente:

- `TVseries.hpp`: definição das classes para representação da aplicação (`TitleBasics`, `TitlePrincipals`, `TitleEpisode` e `TVSeriesAPP`).
- `TVseries.cpp`: implementação dos métodos relativos às classes definidas em `TVseries.hpp`.
- `testTP5.cpp`: inclui o programa principal que invoca e realiza testes básicos às funções implementadas.
- `.tsv`: ficheiros de texto para teste das funções implementadas.

Notas importantes:

1. **Ambos os ficheiros `TVseries.hpp` e `TVseries.cpp` podem ser alterados.**
2. **Cada atributo das classes definidas apresenta detalhes adicionais junto a cada um deles em `TVseries.hpp`.**

O ficheiro contém as classes:

1. `TitleBasics` – caracteriza cada série televisiva,
2. `TitleEpisode` – caracteriza cada episódio de cada série,
3. `TitlePrincipals` – caracteriza quem entra em cada episódio (equipa ou elenco),
4. `TVSeriesAPP` – define a aplicação para gestão das séries televisivas.

Classe `TitleBasics`

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

1) identificador único, alfanumérico, do título (`tconst`)
2) tipo/formato do título (`titleType`); por exemplo, "movie", "short", "tvseries", "tvepisode", "video", etc
3) título mais popular (`primaryTitle`)
4) título original, na língua original (`originalTitle`)
5) booleano que identifica se é para não-adultos – 0 – ou para adultos – 1 – (`isAdult`)
6) ano de lançamento (`startYear`); no caso de uma série, representa o ano de início da série
7) ano final da série (`endYear`); para os outros tipos que não correspondem a "tvseries", é representado com "\N"
8) duração, em minutos (`runtimeMinutes`)
9) vetor de *strings* que inclui, no máximo, três géneros associados ao título (`genres`)

Classe `TitleEpisode`

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

1) identificador único, alfanumérico, do episódio (`tconst`)
2) identificador único, alfanumérico, do título da série a que o episódio pertence (`parentTconst`)
3) temporada a que o episódio pertence (`seasonNumber`)
4) número do episódio da série (`episodeNumber`)

Classe `TitlePrincipals`

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

1) identificador único, alfanumérico, do episódio da série do qual fez parte (`tconst`)
2) número para identificar de forma exclusiva as linhas de determinado título (`ordering`)
3) identificador único, alfanumérico, da pessoa (`nconst`)
4) nome por qual a pessoa é conhecida (`primaryName`)
5) ano de nascimento (`birthYear`)
6) categoria do trabalho (`category`)
7) cargo (`job`); se não for aplicável, é representado com "\N"

8) nome da personagem representada (`characters`); se não for aplicável, é representado com "\N"

Classe `TVSeriesAPP`

A classe `TVSeriesAPP` contém os métodos para representar a aplicação.

As funções a implementar neste trabalho correspondem aos métodos definidos na classe anterior.

Nota Importante

Todas as classes já definidas podem ser manipuladas para adicionar novos atributos e métodos. Novas classes podem também ser implementadas. As estruturas de dados a utilizar na implementação da classe `TVSeriesAPP` devem ser **escolhidas** de modo a obter a melhor eficiência na solução criada.

Os métodos `addTitleBasics`, `addTitleEpisodes` e `addTitlePrincipal` devem ser implementados para utilização nas funções `parseTitleBasics`, `parseTitleEpisodes` e `parseTitlePrincipals`, que permitem preencher as classes definidas de acordo com a informação dos ficheiros de texto.

Classe `TVSeriesAPP`

`Funções auxiliares, que têm de implementar:`

`TVSeriesAPP``();`
Construtor. Cria um objeto da classe `TVSeriesAPP`*.*

`~TVSeriesAPP``();`
Destrutor. Destrói um objeto da classe `TVSeriesAPP`*, apagando-o na memória.*

`void` **`addTitleBasics`**`(const TitleBasics& title);`
Adiciona um novo elemento (título) a `TVSeriesAPP`*.*

`void` **`addTitleEpisodes`**`(const TitleEpisodes& episode);`
Adiciona um novo elemento (episódio associado a determinado título) a `TVSeriesAPP`*.*

`void` **`addTitlePrincipal`**`(const TitlePrincipal& principal);`
Adiciona um novo elemento (pessoa associada a determinado episódio de determinada série) a `TVSeriesAPP`*.*

As seguintes funções são as funções que serão avaliadas e também serão avaliadas as suas eficiências:

1. `vector<string>` **`getUniquePrincipals`**`(const string& seriesTconst) const;`

Encontra todas as pessoas que entraram em episódios de determinada série, ou seja, pesquisa todos os elementos diferentes da classe `TitlePrincipals` *da série de ID* `seriesTconst`*. Retorna o vetor com o nome das pessoas (*`primaryName`*) encontradas, organizado alfabeticamente. Em caso de erro, retorna um vetor vazio.*

2. `string` **`getMostSeriesGenre`**`() const;`

Encontra o género mais frequente das séries na classe `TVSeriesAPP`*. No caso de vários géneros terem o mesmo número de séries associadas, é escolhido o que tiver o nome com menos caracteres. Retorna o género encontrado, ou uma string vazia em caso de erro.*

3. `vector<string>` **`principalsWithMultipleCategories`**`(const string& seriesTconst) const;`

Encontra todas as pessoas que desempenharam diferentes categorias no trabalho desenvolvido nos episódios em que entraram de determinada série de ID `seriesTconst`*. Retorna o vetor com o nome das pessoas (*`primaryName`*) encontradas, organizado alfabeticamente. Em caso de erro, retorna um vetor vazio.*

4. `vector<string>` **`principalsInAllEpisodes`**`(const string& seriesTconst) const;`

Encontra todas as pessoas que entraram na totalidade de episódios de determinada série de ID `seriesTconst`*. Retorna o vetor com o nome das pessoas (*`primaryName`*) encontradas, organizado alfabeticamente. Em caso de erro, retorna um vetor vazio.*

5. `int` **`principalInMultipleGenres`**`(vector<string> vGenres) const;`

Determina o número de pessoas que entraram em séries com géneros correspondentes aos géneros em `vGenres`*, retornando-o.*

6. `string` **`getPrincipalFromCharacter`**`(const string& character) const;`

Encontra a pessoa que representou mais vezes determinada personagem (`character`*) nos episódios. No caso de a contagem resultar em empate, é selecionada a que se encontra alfabeticamente à frente. Retorna o nome da pessoa (*`primaryName`*) encontrada, ou uma string vazia em caso de erro.*

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 `testTP5.cpp`. 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 `testTP5`, quando executado, deverá apresentar o seguinte resultado:

```
INICIO DOS TESTES

Base de dados pequena (5 series)

...verifica_getUniquePrincipals5: (Serie - Voetbal Inside) - Os elementos (=Chris Woerts, Danny Vera, Gertjan Verbeek, Hans Kraay Jr., Jan Boskamp, Johan Derksen, Johnny de Mol, Marcel van Roosmalen, René van der Gijp, Roelof Luinge, Ronald Koeman, Sensi
```

```
Lowe, Simon Zijlemans, Valentijn Driessen, Walter Trout, Wilfred Genee, Willem Feenstra, Wim Kieft) são os esperados (ok)
...verifica_getUniquePrincipals5: (Serie - Weird Norwegian) - Os elementos (=Albert Cerný, Antonín Hrabal, Jeronym Subrt, Victor Sotberg) são os esperados (ok)
OK: verifica_getUniquePrincipals5 passou

...verifica_getMostSeriesGenre5: (Genero com mais series) - A serie (=Talk-Show) é o esperado (ok)
OK: verifica_getMostSeriesGenre5 passou

...verifica_principalsWithMultipleCategories5: (Serie 'Voetbal Inside') - Não existe nenhuma pessoa (ok)
...verifica_principalsWithMultipleCategories5: (Serie 'Wish ko lang') - As pessoas (=Jeffrey Hidalgo) são o esperado (ok)
OK: verifica_principalsWithMultipleCategories5 passou

...verifica_principalsInAllEpisodes5: (Serie: Voetbal Inside) - As pessoas (=Johan Derksen, René van der Gijp, Wilfred Genee) são o esperado (ok)
...verifica_principalsInAllEpisodes5: (Serie: Wish ko lang) - As pessoas (=) são o esperado (ok)
OK: verifica_principalsInAllEpisodes5 passou

...verifica_principalInMultipleGenres5: (Genero: Sport, Talk-Show) - O total (=18) são o esperado (ok)
...verifica_principalInMultipleGenres5: (Genero: Crime, Documentary, Drama) - O total (=175) são o esperado (ok)
OK: verifica_principalInMultipleGenres5 passou

...verifica_getPrincipalFromCharacter5: (Pessoa que interpetou de 'Self' mais vezes) - A pessoa (=Johan Derksen) é o esperado (ok)
...verifica_getPrincipalFromCharacter5: (Pessoa que interpetou de 'Raul' mais vezes) - A pessoa (=Kristof Garcia) é o esperado (ok)
OK: verifica_getPrincipalFromCharacter5 passou

Fim dos testes da base de dados pequena



Base de dados grande

...verifica_getUniquePrincipals: (Serie - Corruption) - Os elementos  (=Alex Butcher, Andrew Katz, Anthony Rosario, Brett Smith, Curtis Maloney, Dan Kelly, David Kern, Emily Dunlop, Jack Dalton, Jack Moynihan, Jenna Phelan, John DiSabito, John O'Connor, Katelyn Johnson, Leo Dibbern, Mahak Kanjolia, Matt O'Brien, Rachel Michaelson, Rose Stella, Ryan Imelio, Sammie Dibbern, ) são os esperados (ok)
OK: verifica_getUniquePrincipals passou

...verifica_getMostSeriesGenre: (Genero com mais series) - A serie (=Talk-Show) é o esperado (ok)
OK: verifica_getMostSeriesGenre passou

...verifica_principalsWithMultipleCategories: (Serie Ted and Johnny) - Não existe nenhuma pessoa (ok)
...verifica_principalsWithMultipleCategories: (Serie 'Call Me Katie') - As pessoas (=Chris Greenwood, Dave Baines, David Foulkes, Di Evans, Donnovan Harris, Joseph Hassell, Laura Balfour, Matt Jackson, Matt Ward) são o esperado (ok)
OK: verifica_principalsWithMultipleCategories passou

...verifica_principalsInAllEpisodes: (Serie: Lakshmi Vanthachu) - As pessoas (=S. Sekilar, Saran Rajesh, Suresh Krishna, Vani Bhojan) são o esperado (ok)
```

```
...verifica_principalsInAllEpisodes: (Serie: Secret Diaries) - As pessoas (=Divya Aggarwal, ) são o esperado (ok)
OK: verifica_principalsInAllEpisodes passou

...verifica_principalInMultipleGenres: (Genero: Sport, Talk-Show) - O total (=52) são o esperado (ok)
...verifica_principalInMultipleGenres: (Genero: Crime, Documentary, Drama) - O total (=97) são o esperado (ok)
OK: verifica_principalInMultipleGenres passou

...verifica_getPrincipalFromCharacter: (Pessoa que interpetou de 'Self' mais vezes) - A pessoa (=Aaron Elliott) é o esperado (ok)
...verifica_getPrincipalFromCharacter: (Pessoa que interpetou de 'Judy' mais vezes) - A pessoa (=Elora Españo) é o esperado (ok)
OK: verifica_getPrincipalFromCharacter passou

FIM DOS TESTES: Todos os testes passaram
```

Sugestão: Existem duas bases de dados para testarem as funções (uma com 5 series outra com 1000). Sugerimos na implementação das funções para não se perder muito tempo, usem só a de 5 series. Para isso comentem no ficheiro **testTP5.cpp** todos os testes que testem a base de dados de 1000. No ficheiro está indicado o local onde devem começar a comentar (*/*) e o local onde devem terminar o comentário (*/*)

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.

6. Avaliação

A classificação do trabalho é dada pela avaliação feita à implementação submetida pelos estudantes, considerando ainda a **eficiência** da resolução. A classificação final do trabalho (TP5) é dada por:

$$TP5 = (0.75\ Implementação + 0.25\ Eficiência) \times AO$$

A classificação da implementação é essencialmente determinada por **testes automáticos** adicionais (por exemplo, recorrendo a ficheiros de teste de maiores dimensões). No caso de a implementação submetida não compilar, esta componente será 0%.

A **eficiência** das soluções submetidas será avaliada tendo em consideração os **tempos** de todas as implementações submetidas pelos estudantes, que serão organizadas por ordem crescente de tempo e divididas em 5 patamares: 100% – primeiros 20%; 80% – segundos 20%; 60% – terceiros 20%; 40% – quartos 20%; 20% – últimos 20%.

A avaliação oral (AO) será dividida em 4 patamares: 100% – domina o código; 75% – algumas falhas; 40% – várias falhas detetas na explicação; 0% – demonstrou graves lacunas.

7. Teste em servidor

Em breve, será disponibilizado um servidor para que o código possa ser testado durante o desenvolvimento. O código submetido neste servidor **NÃO SERÁ AVALIADO**. Apenas a submissão via Moodle é válida para efeitos de avaliação.

8. Submissão da resolução

A submissão é apenas possível através do Moodle e até à data indicada no início do documento. Deverá ser submetido um ficheiro *zip* contendo:

- os ficheiros `TVseries.hpp` e `TVseries.cpp` com as funções implementadas;
- um ficheiro `autores.txt` indicando o nome e número dos elementos do grupo.

Nota importante: Apenas as submissões com o seguinte nome serão aceites: `T5_G<numero_do_grupo>.zip`. Por exemplo: `T5_G9999.zip`.

9. Plágio

As implementações submetidas serão analisadas num programa para deteção de plágio. Quaisquer cópias identificadas serão devidamente penalizadas.