e-fólio B

Tendo como base o conteúdo teórico apresentado nos tópicos estudados até agora, implemente um programa em C++, que consiga atender ao *briefing* solicitado abaixo. O trabalho deverá ser carregado em área apropriada na plataforma, enviando o código-fonte e o relatório descritivo e justificativo das opções tomadas para solucionar o problema.

Enviar o trabalho em ficheiro comprimido com nome atendendo o seguinte formato: PrimeiroNomeÚltimoNome_NúmeroMatrícula_efolioA.zip.

O trabalho é individual e qualquer evidência de cópia levará à anulação dos trabalhos envolvidos.

O trabalho será avaliado segundo as seguintes diretrizes:

Critérios de exclusão:

- ◎ Não compilar ou executar
- ◎ Não executar o mínimo pretendido

Critérios de inclusão:

- ◎ Fidelidade ao enunciado – de 7 até 12 valores
- ◎ Orientação a objetos bem desenvolvida – até 3 valores
- ◎ Simplicidade e legibilidade – até 2 valores
- ◎ Eficiência e inovação – até 2 valores
- ◎ Qualidade do relatório – até 1 valor

A nota final será convertida para uma escala de 0-4 valores.

Briefing:

No primeiro eFolio trabalharam sobre uma estrutura de dados em formato de folha de cálculo e dedicaram-se à extração de classes emergentes dos dados. No segundo eFolio o briefing mantém a liberdade na escolha do tema e dos dados, mas aumenta a complexidade para responder a uma das questões essenciais de POO: a herança.

Pretende-se que construam um pequeno programa para exploração de dados em estrutura hierárquica/arborescente. Um formato interessante para esta estrutura é o XML e há vários repositórios online. Exemplos de estruturas deste género são as árvores taxonómicas ou cladísticas em biologia, organização de corpos celestes em cosmologia, ou mesmo organização de categorias no catálogo online da Amazon. Convém não escolherem um problema excessivamente grande ou implementarem apenas um pequeno subconjunto. Ao contrário do primeiro eFolio relativamente ao CSV, não é relevante o tratamento do XML ou sequer que haja tratamento XML caso queiram ter uma pequena estrutura directamente no código, e podem inclusivamente montar manualmente os vossos dados de várias fontes. O tempo de execução será mais bem gasto a perseguir os requisitos pedidos de uma forma bem desenhada:

- o utilizador deve poder explorar as propriedades das classes navegando nos vários níveis hierárquicos.

- deve haver pelo menos um exemplo de **herança simples** e um **método overridden** (pensem bem na escolha de tema em função destes requisitos).

- o relatório deve explicitar como, onde e porquê implementaram os dois itens do ponto anterior.

- escusado será dizer que devem respeitar os paradigmas de POO vistos no anterior eFolio, incluindo separação entre ficheiros .h e .cpp, respeito por princípios de encapsulação, atenção a construtores/destrutores, etc…

- devem manter o código explícito, legível, com nomes claros de variáveis e comentários informativos, para vós próprios e para o leitor.

- devem balizar o projeto para que não seja demasiado minimalista nem demasiado complexo, um mínimo de 4 e máximo de 8 classes será em princípio o expectável, e em redor das 500 linhas de código de implementação (código a metro é uma péssima ideia, só para vos dar uma ideia de que é algo da dimensão aproximada do que apresentaram no eFolio anterior, que teria menos variabilidade na abordagem, mas considerem um intervalo muito largo para este valor).

Não esquecer a entrega de todos os elementos relevantes para a aferição da qualidade do projeto, incluindo o código, relatório, executável e ficheiro de dados (quando aplicável).