Departamento de Matemática — Universidade de Coimbra

Ano Lectivo de 2016/2017 | Programação Orientada para os Objectos | Frequência 27/04/2016

Duração da prova: 2h30

---

1. A *Modularidade* pode ser descrita como sendo a propriedade de um sistema que foi decomposto num conjunto de módulos coerentes e interligados entre si. Como é que, numa linguagem de *Programação Orientada para os Objectos*, o conceito de *Modularidade* é posto em prática?

2. Numa relação de herança qual é o significado das secções `public`, `protected` e `private` na *classe de Base* e nas *classes Derivadas*?

3. Quais são os tipos básicos de relações entre classes em *C++*? Para cada um dos casos apresente um diagrama UML exemplificativo.

4. Os concessionários da eUMM pretendem construir um programa que permita aos eventuais clientes recorrerem a uma simulação (em termos de escolha do modelo+extras e do preço final) antes de decidirem que modelo e que extras é que vão escolher.

   (a) Construa um diagrama de classes (UML) apropriado à manipulação da informação respeitante a: carros, componentes (tipos de motores, jantes, estofos, etc.), e extras (GPS, RádioMP3, ABS, etc.). Tenha o cuidado de considerar os atributos mais importantes para o problema em questão.

   (b) De entre as diferentes classes que constituem o seu sistema, escolha uma e, para essa classe, apresente a sua especificação (ficheiro `.hpp`)

5. Método de Ordenação *Fusão*

   Dados dois vectores, ordenados, podemos fundi-los de forma ordenada num outro vector.

   - Comparam-se os dois elementos iniciais dos dois vectores.
   - Escolhe-se o vector que contêm o elemento menor e copiam-se os seus elementos para o vector final até que o elemento que se está a considerar já seja maior do que o elemento inicial do outro vector.
   - troca-se de vector e repete-se o processo, até que se chega ao fim de um dos vectores.
   - copiam-se os restantes elementos do outro vector para o vector final.

   O processo precisa sempre de um vector auxiliar, sobre o qual se vai construir o vector ordenado.

   Pode-se ordenar um vector considerando que o mesmo é constituído por dois sub-vectores, isto é, divide-se o vector inicial em dois, fazendo de seguida a fusão ordenada desses dois sub-vectores.

   Dado que as metades iniciais não estarão, provavelmente, ordenadas, repete-se (de forma recorrente) o processo de divisão até se atingir vectores individuais, que estão, obviamente, ordenados. O processo de fusão ordenada começa então a partir daí até se atingir o vector final.

   (a) Construa uma classe abstracta *Ordenação* com um método (também genérico) que implemente o método de *ordenação por fusão*. Use a classe `list` da biblioteca padrão do *C++*, assim como os iteradores correspondentes.

   (b) Explique o que são os *Iteradores* e qual é a sua necessidade? Explique porque é que não é possível recorrer directamente aos ponteiros?
