

**Relatório do Trabalho de Algoritmos e Estruturas de Dados**

*Gestão de Bilhetes no Metro*

**2018**

por

João Pedro Ribeiro | 201704851 | up201704851@fe.up.pt

Rui Teixeira | 201707001 | up201707001@fe.up.pt

Vítor Ventuzelos |201706403 | up201706403@fe.up.pt

1 | Tema do Trabalho

Este relatório versa sobre a resolução da primeira parte do trabalho da cadeira de Algoritmos e Estruturas de Dados na Faculdade de Engenharia da Universidade do Porto, realizado no primeiro semestre do ano letivo 2018/19, pelos alunos do grupo seis da turma três do segundo ano do Mestrado Integrado em Engenharia Informática e Computação: João Pedro Ribeiro (201707851), Rui Teixeira (201707001) e Vítor Ventuzelos (201706403).

O nosso trabalho tem como tema o tema quatro – Gestão de Bilhetes no Metro – dos temas sugeridos. O objetivo do trabalho foi resolver um problema recorrendo ao paradigma da programação orientada por objetos e utilizando a linguagem C++.

O problema passava pela implementação de um sistema de gestão informática da bilhética e da utilização do Metro do Porto. Assim, era necessário processar uma série de informações sobre o metro (relativo a tarifas, tipos de assinatura e zonas de circulação) para fazer uma correta implementação da solução.

O sistema de gestão informática deveria ter todas estas informações para que fosse possível conhecer os valores e tipos de bilhetes vendidos em qualquer um dos vários pontos de venda ou máquinas automáticas da rede de venda da Metro do Porto bem como conhecer os viajantes (guardando os seus dados) associados às assinaturas.

2 | Solução Implementada

A solução implementada para resolver o problema proposto (sucintamente descrito na secção anterior) consistiu na implementação de três classes diferentes: a classe **Bilhete**, a classe **PostoDeVenda** e a classe **MaqAutomatica**, uma subclasse da classe PostoDeVenda.

Na **main** implementamos uma série de funções necessárias ao normal funcionamento do programa, respondendo às necessidades previstas:

1. A função **procurarPosto** que permite a procura de um posto de venda ou de uma máquina automática através da localização do mesmo.
2. A função **adicionaPosto** que permite adicionar um posto de venda ou uma máquina automática inserindo a localização do mesmo.
3. A função **operaMaquina** que permite, em cada máquina, adicionar um bilhete, ver os bilhetes que foram vendidos e ordenar os bilhetes vendidos nesse ponto até ao momento, segundo critérios fixos de ordenação.
4. A função **addBilheteMaq** que permite adquirir um bilhete único ou diário para qualquer uma das zonas geográficas, emitindo o bilhete e registando na respetiva máquina essa venda.
5. A função **addBilhetePosto** que permite emitir um bilhete num posto, tal como nas máquinas, permitindo, no entanto, adquirir uma assinatura mensal do tipo normal, júnior, sénior ou estudante (algo que as máquinas automáticas estão impossibilitadas de fazer), recolhendo, para o efeito, uma série de dados necessários: nome, cartão de cidadão, idade e, no caso da assinatura ser do tipo estudante, escola.

Para que seja possível a utilização destas funções na **main** são, obviamente, usados membros-função das classes **Bilhete, PostoDeVenda** e **MaqAutomatica**.

Cada posto de venda tem como elemento distintivo dos outros postos de venda a sua localização (do tipo *string*). A cada posto de venda está ainda associado um *vector* com a informação sobre os bilhetes emitidos nesse mesmo ponto de venda (*vector<Bilhete> bilhetes\_emitidos*). A classe **PostoDeVenda** tem membro-função funções básicas que retornam os bilhetes já emitidos, o número de bilhetes já emitidos, a localização do posto de venda, uma função que ordena os bilhetes usando operadores de comparação, uma função que emite o bilhete e uma função que imprime os bilhetes emitidos no posto de venda (todos os bilhetes emitidos ou apenas durante um intervalo de tempo, sendo que esta última funcionalidade ficou com um erro não identificado) Cada máquina automática tem também como elemento caraterístico a sua localização (do tipo *string*).

O construtor de bilhetes na classe Bilhete recebe a sua categoria, o tipo de bilhete, o dia, mês e ano em que é emitido e, caso se trate de uma assinatura, os elementos necessários: nome, idade, número do cartão de cidadão e idade. A classe **Bilhetes** tem como membro-função funções básicas que devolvem a categoria, tipo, preço, escola, nome, cartão de cidadão e idade do proprietário, bem como uma função que imprime os bilhetes já existentes e dois operadores de comparação entre os bilhetes.

3 | Diagrama UML

![Uma imagem com captura de ecrã

Descrição gerada automaticamente]()

4 | Lista de Casos de Utilização

Como casos de utilização possível para a solução proposta para o problema, há a perspetiva de utilização da gestão do Metro do Porto e a perspetiva de utilização do viajante.

Do ponto de vista da gestão do Metro do Porto, o programa pode ser utilizado para controlar a rede de postos ou máquinas de venda automáticas que a empresa dispõe e utiliza, adicionando ou vendo informações sobre cada um dos postos ou das máquinas automáticas. Sobre cada máquina, a empresa consegue utilizar o programa para ver os bilhetes vendidos, durante todo o período de venda ou durante um intervalo de tempo específico, e ordená-los.

Do ponto de vista do viajante do metro, o programa pode ser utilizado para registar num determinado posto de venda ou máquina automática a emissão de um determinado tipo de bilhete (único ou diário) ou de assinatura (normal, júnior, sénior ou estudante), respeitando que as máquinas automáticas apenas emitam bilhetes e que apenas os postos de venda possam emitir assinaturas.

Apesar das especificidades deste programa para a resolução do problema específico da Metro do Porto, ele poderá ser facilmente adaptado para a utilização numa qualquer outra empresa para gestão de questões semelhantes.

5 | Principais dificuldades encontradas no desenvolvimento do trabalho

Durante o desenvolvimento do trabalho não foram sentidas grandes dificuldades na implementação da solução para o problema dado, com exceção da existência de um erro não identificado na funcionalidade que permite a impressão dos bilhetes emitidos num determinado posto de venda num determinado intervalo de tempo, na função *printBilhetes* da classe **PostoDeVenda**.

6 | Esforço Dedicado

Esforço dedicado em termos percentuais por elemento do grupo:

* João Pedro Ribeiro: 30%
* Rui Teixeira: 15%
* Vítor Ventuzelos: 55%