

**algorimtos e estruturas de dados**

**MIEIC**

**Turma 2 – Grupo D**

Inês Teixeira – [up201404592@fe.up.pt](mailto:up201404592@fe.up.pt)

José Aleixo da Cruz – [up201403526@fe.up.pt](mailto:up201403526@fe.up.pt)

José Miguel da Costa – [up201402717@fe.up.pt](mailto:up201402717@fe.up.pt)

Dezembro 2015

Relatório de Projeto

Empresa de transporte de mercadorias

# Tema de trabalho

O nosso grupo trata do desenvolvimento de um programa em C++, que tem como propósito lidar com a informação de uma empresa de transporte de mercadorias.

No início, desenvolvemos métodos que se debruçavam sobre os clientes, serviços e meios de transporte da empresa, que apelidámos caridosamente de “Transportex”. Desta vez, coube-nos elaborar estruturas de dados que guardassem informação relativa a motoristas, reparações e clientes inativos desta empresa.

No âmbito da cadeira de Algoritmos e Estruturas de Dados (AEDA) foram lecionados três tipos importantes de contentores de dados: árvores binárias, filas de prioridade e tabelas de dispersão. É fazendo uso destes contentores que desenvolvemos a solução para esta segunda parte do projeto.

# Solução implementada

***Binary search tree***

A partir da definição de uma árvore de pesquisa binária (BST), presente no ficheiro *header* disponível no Moodle, desenvolvemos métodos capazes de adicionar, editar, atualizar e remover motoristas da empresa de uma BST.

Na classe “Empresa”, que foi elaborada na primeira parte do trabalho, adicionamos um membro privado denominado “motoristas”, que representa uma árvore binária de uma nova classe “Motorista”, cujo critério de ordenação é o número de horas de trabalho efetuado pelo motorista num determinado dia (ordem crescente).

A classe “Motorista” tem como atributos privados o nome, número de identificação fiscal (NIF) e horas de serviço diário já efetuado do motorista. Desenvolvemos o respetivo construtor e as funções que alteram estes atributos.

Sempre que um motorista da árvore binária é alterado, de forma a manter a ordenação da árvore, removemos os dados desse motorista da árvore, atualizamos os dados e voltamos a inseri-lo.

Temos funções que fazem uma listagem dos 10 motoristas com menos e mais horas de trabalho, assim como listagem de todos os motoristas e “reset” das horas de trabalho diárias.

**Fila de prioridade**

Utilizando o contentor *priority\_queue* fornecido pela *standard library* do C++, gerámos uma fila de prioridades com o objetivo de controlar as eventuais reparações que os camiões da empresa necessitam de efetuar, atendendo à disponibilidade das oficinas e ao tipo de serviços (específicos ou casuais) que oferecem.

**Tabela de disperção**

Para ajudar a secção de Marketing da Transportex, mantemos a informação relativa a clientes inativos (que não requisitam serviços da empresa há mais de 1 ano) atualizada numa tabela de dispersão, sob a forma de *unordered\_set*.