

Tema 5 – Empresa de Transporte de Mercadorias

- Caio Nogueira;
- Diogo Sousa;
- Miguel Silva.

Empresa de Transportes

Empresa de transporte de mercadorias. A empresa tem um conjunto de camiões específicos para cada tipo de mercadoria (Congelação, Animal, Normal ou Perigosa). Tem também um conjunto de serviços que fornece e que podem ser escolhidas pelos clientes da empresa.

Os funcionários da empresa são motoristas, que conduzem um ou mais tipos de camiões.

A empresa disponibiliza para os seus clientes diferentes serviços de transporte, que utilizam um ou mais camiões cada um.

Os clientes da empresa são identificados pelo nome e NIF, podendo adquirir diferentes serviços de transporte.

Os camiões também necessitam de serviços de manutenção, prestados pelas oficinas, as quais se especializam em determinadas marcas.

Uso de estruturas não lineares

Nesta segunda parte do trabalho, passamos de usar vetores para guardar informação de clientes e de motoristas, para usar estruturas de dados não lineares.

Para o caso dos motoristas, passamos a usar uma BST (binary search tree), ordenando-os pelo número de horas diárias efetuadas.

No caso dos clientes, criamos uma hash table para guardar a informação dos clientes inativos, mantendo os clients ativos num vetor. Sempre que haja uma requisição de serviços ou na passagem de um dia, estas estruturas de dados são atualizadas.

Para guardar a informação das oficinas, criou-se uma priority queue, estas são guardadas por ordem crescent da disponibiliade.

Operadores para o uso da priority queue

```
bool Oficina::operator==(const Oficina &o1) {return this->getNome() == o1.nome && this->disponibilidade == o1.disponibilidade;}  
bool Oficina::operator<(const Oficina &o1) const {return this->disponibilidade > o1.disponibilidade;}
```

Uso de estruturas não lineares

Operadores para BST

```
bool Motorista::operator==(const Motorista m) {
    return this->getName() == m.getName() && this->age == m.age;
}

bool Motorista::operator<(const Motorista& m) const {
    if (this->getTotalHoras() == m.getTotalHoras())
        return this->getName() < m.getName();
    return this->total_horas < m.total_horas;
}
```

Criação de uma struct para a hash table dos clientes inativos

```
struct inactiveClientHash {
    int operator()(const Cliente& c1) const {
        return 0;
    }

    bool operator()(const Cliente& c1, const Cliente& c2) const {
        return c1.getNif() == c2.getNif();
    }
};

typedef unordered_set<Cliente, inactiveClientHash, inactiveClientHash> HashTabInactiveClient;
```

Estrutura de Ficheiros

empresa.txt

```
Transportes EuroTruck
./clientes.txt
./motoristas.txt
./camioes.txt
./distancias.txt
./servicos.txt
```

clientes.txt

```
Rui Almeida
65
123456789
2 ; 3 ; 9
:::::
Caio Nogueira
19
257115412
1
:::::
Jose
```

Nome
Idade
NIF
Serviços encomendados
Cliente ativo/inativo (1/0)
Sep (::::)
Novo cliente

motoristas.txt

```
Caio Nogueira
33
2300
09:00 -> 10:00
n
:::::
Mike Silva
20
2300
10:00 -> 11:00
n
:::::
Quicirus
```

Nome
Idade
Salário
Horário
Categorias
Sep (::::)
Novo motorista

camioes.txt

```
16
p
1
Tesla
1500
8000
toxico
:::::
a
2
Renault
5500
23
100
:::::
n
3
Range Rover
2000
67
:::::
```

(Nº de camiões)
Tipo de camião
ID
Marca
Capacidade
Km percorridos
Especial para cada tipo
Sep (::::)
Novo camião

servicos.txt

```
1
perigosos
6
Porto
Braga
11:00
04/01/2020
1
2
:::::
2
congelacao
9
Braganca
Viana do Castelo
16:00
24/03/2020
1
5
```

ID
Categoria da encomenda
Camiões para entrega
Origem
Destino
Hora
Data
Disponibilidade
Horas de viagem
Sep (::::)
Novo serviço

distancias.txt

```
Porto -> Aveiro - 75
Porto -> Braga - 54.3
Porto -> Vila Real - 95.0
Porto -> Viseu - 127.9
Porto -> Braganca - 207.2
Porto -> Viana do Castelo - 74.4
Porto -> Guimaraes - 55.4
Aveiro -> Braga - 123.9
Aveiro -> Viseu - 84.6
Aveiro -> Braganca - 269.8
Aveiro -> Vila Real - 157.1
Aveiro -> Guimaraes - 73.2
Aveiro -> Viana do Castelo - 153.0
Viseu -> Braganca - 198.0
Viseu -> Braga - 176.2
Viseu -> Guimaraes - 177.0
Viseu -> Vila Real - 91.0
Viseu -> Viana do Castelo - 199.7
Guimaraes -> Braga - 131.1
Guimaraes -> Braganca - 198.0
Guimaraes -> Viana do Castelo - 81.2
Guimaraes -> Vila Real - 85.8
Braga -> Braganca - 205.8
```

Oficinas.txt

```
Queiros Mecanico
Volkswagen
2
04/01/2020
:::::
Aunt Dulce
Fiat ; SEAT ; Mercedes
2
04/01/2020
```

Nome
Marca especializada
Dias até estar disponível
Data até estar disponível
Sep(::::)
Nova oficina

Funcionalidades Implementadas

- ◇ Como já referido anteriormente, foram implementadas no programa todas as funcionalidades que permitem adição, remoção e edição de objetos das classes Camiao, Cliente, ServicoTransporte e Motorista.
- ◇ Para além disso, é possível também visualizar toda a informação da empresa: acerca dos seus clientes, motoristas, camiões (overload dos operadores de extração), e informação das oficinas, bem como a informação financeira (mensal) da empresa.

Empresa.h

```
/**  
 * @brief Le informação de clientes no respetivo ficheiro e guarda a informação na base de dados  
 *  
 */  
void readClientes();  
  
/**  
 * @brief Le informação de motoristas no respetivo ficheiro e guarda a informação na base de dados  
 *  
 */  
void readMotoristas();  
  
/**  
 * @brief Le informação de camiões no respetivo ficheiro e guarda a informação na base de dados  
 *  
 */  
void readCamioes();
```

Menus.h

```
/**  
 * @brief Sub-menu do menu principal, onde se pode aceder às funções que dizem respeito a estatísticas  
 *  
 */  
void Estatisticas(Empresa& e1);  
  
/**  
 * @brief Menu que permite gerir os serviços das oficinas que são prestados aos camiões  
 *  
 */  
void gerirServicosOficinas(Empresa& e1);  
  
/**  
 * @brief Menu principal, onde se pode aceder a vários sub-menus para operar sobre um detalhe particular da empresa  
 *  
 */  
* @param e1 Base de dados para passar às funções que operam sobre esta  
* @return unsigned 1 se escolher a opção para sair do programa, 0 para qualquer outra opção  
*/  
unsigned mainMenu(Empresa& e1);
```

Objetivos Globais

- ❖ Relativamente aos objetivos do projeto no geral, consideramos que estes foram cumpridos, dado que se encontram funcionais no programa, sendo assim possível ao utilizador realizar operações de CRUD.

```
|| ====== MAIN MENU ======
|| (0) Sair
|| (1) Gerir clientes
|| (2) Gerir camioes
|| (3) Gerir servicos de transporte
|| (4) Gerir motoristas
|| (5) Mostrar estatisticas da empresa
|pcao:
```


Outras funcionalidades

Para além das funcionalidades já referidas anteriormente, implementamos também algumas funções e métodos que permitiram auxiliar a organização da informação e, de modo geral, a resolução do problema dado.

Uma dessas funcionalidades é o método público `getPreco()` da classe `ServicoTransporte`. O preço de cada serviço de transporte é, deste modo, calculado a partir da distância entre a origem e o destino, sendo que esta informação se encontra guardada no ficheiro `distancias.txt`.

O preço tem em conta também outras características do serviço, como o número de camiões necessários e o tipo de mercadorias transportadas.

```
double ServicoTransporte::getPreco(map<pair<string,string>,double > distancias) const {  
    double result = this->getDistancia(distancias)*this->getCamioes().size();  
    if (this->getTipo() == "perigosos") result *= this->PerigososMultiplicador;  
    else if (this->getTipo() == "congelacao") result *= this->CongelacaoMultiplicador;  
    else if (this->getTipo() == "normal") result *= this->NormalMultiplicador;  
    else if (this->getTipo() == "animais") result *= this->AnimaisMultiplicador;  
    return result;  
}
```

```
Porto -> Aveiro - 75  
Porto -> Braga - 54.3  
Porto -> Vila Real - 95.0  
Porto -> Viseu - 127.9  
Porto -> Braganca - 207.2  
Porto -> Viana do Castelo - 74.4  
Porto -> Guimaraes - 55.4  
Aveiro -> Braga - 123.9  
Aveiro -> Viseu - 84.6  
Aveiro -> Braganca - 269.8  
Aveiro -> Vila Real - 157.1  
Aveiro -> Guimaraes - 73.2  
Aveiro -> Viana do Castelo - 153.0  
Viseu -> Braganca - 198.0  
Viseu -> Braga - 176.2  
Viseu -> Guimaraes - 177.0  
Viseu -> Vila Real - 91.0  
Viseu -> Viana do Castelo - 199.7  
Guimaraes -> Braga - 131.1  
Guimaraes -> Braganca - 198.0  
Guimaraes -> Viana do Castelo - 81.2  
Guimaraes -> Vila Real - 85.8  
Braga -> Braganca - 205.8  
Braga -> Vila Real - 105.3  
Braga -> Viana do Castelo - 61.5  
Braganca -> Vila Real - 118.2
```

Funcionalidade em Destaque

- ❖ Uma das funcionalidades extra (não exigidas pelo enunciado do problema), é o horário de trabalho dos funcionários e dos serviços de transporte, que têm que coincidir para que o serviço de transporte seja bem sucedido, isto é, para que um serviço de transporte se possa realizar, é necessário que exista pelo menos um motorista devidamente preparado, cujo horário de trabalho abranja o horário marcado para o serviço de transporte. Caso isto não aconteça, é lançada a exceção MotoristasIndisponiveis.

```
bool Motorista::isWorking(string time) const {
    istringstream str(time);
    string token;
    istringstream inicio(horario.first);
    istringstream fim(horario.second);
    return compareTime(str, inicio, fim);
}
```

```
'tempo = hora*60*60 + minutos*60;
tempo_inicio = hora_inicio*60*60 + minutos_inicio*60;
tempo_fim = hora_fim*60*60 + minutos_fim*60;

if (hora_inicio > hora_fim && tempo <= (23*60*60 + 59*60) && tempo >= tempo_inicio)
    return 1;
else if (hora_inicio > hora_fim && tempo >= 0 && tempo <= tempo_fim)
    return 1;
else return tempo_inicio <= tempo && tempo <= tempo_fim;
}'}
```

Principais Dificuldades

- ❖ Ao longo do projeto, foram surgindo diversos erros e bugs.
- ❖ Um dos principais problemas foi o acesso a informações específicas de cada classe derivada de Camiao, quando acedíamos a esse objeto através do vetor de apontadores de camiões. A solução encontrada foi o uso do `dynamic_cast`.
- Outra dificuldade encontrada foi a organização e distribuição da informação nos ficheiros de texto.
- Para além disso, o enunciado parecia-nos um pouco vago, o que acabou por nos dar mais liberdade no desenvolvimento do projeto, mas que, no início, nos deixou um pouco confusos sobre como deveriam ser implementados diversos objetivos.

```
Perigosos *np = dynamic_cast<Perigosos*>(c);
Normal* nn = dynamic_cast<Normal*>(c);
Animais* na = dynamic_cast<Animais*>(c);
Congelacao* nc = dynamic_cast<Congelacao*>(c);
```

Participação dos elementos do grupo

- ❖ Durante a realização do projeto, conseguimos estabelecer uma organização adequada, de modo que todos os elementos do grupo contribuíram de igual forma para o resultado final alcançado.