

**1ª aula prática - Introdução ao CLion e testes unitários. Classes e vetores.****Instruções**

- Faça download do ficheiro *aed2122\_p01.zip* da página da disciplina e descomprima-o (contém a pasta *lib*, a pasta *Tests* com os ficheiros *parque.h*, *parque.cpp* e *tests.cpp*, e os ficheiros *CMakeLists* e *main.cpp*)
- No CLion, abra um **projeto**, selecionando a pasta que contém os ficheiros do ponto anterior.
- Efetuar “Load CMake Project” sobre o ficheiro *CMakeLists.txt*
- Execute o projeto (**Run**)
- Note que os *seis testes unitários deste projeto estão comentados*. Retire os comentários à medida que vai implementando os testes.
- *Deverá realizar esta ficha respeitando a ordem das alíneas.*
- Efetue a implementação no ficheiro *parque.cpp*.

**Enunciado**

Pretende-se implementar um programa para gestão de um parque de estacionamento, que deve gerir a informação sobre os clientes e estacionamento das respetivas viaturas. Implemente a classe *ParqueEstacionamento* de acordo com as alíneas seguintes. A declaração da classe deve ser feita no ficheiro *Parque.h* e a definição dos seus membros-função no ficheiro *Parque.cpp*.

```
class InfoCartao {  
public:  
    string nome;  
    bool presente;  
};
```

```
class ParqueEstacionamento {  
    unsigned vagas;  
    const unsigned lotacao;  
    vector<InfoCartao> clientes;  
    const unsigned numMaximoClientes;  
public:  
    ParqueEstacionamento(unsigned lot, unsigned nMaxCli);  
    bool adicionaCliente(const string & nome);  
    bool retiraCliente(const string & nome);  
    bool entrar(const string & nome)  
    bool sair(const string & nome);  
    int posicaoCliente(const string & nome) const;  
    unsigned getNumLugares() const;  
    unsigned getNumMaximoClientes() const;  
    unsigned getNumLugaresOcupados() const;  
    unsigned getNumClientesAtuais() const;  
};
```

- a) Implemente o construtor da classe **ParqueEstacionamento**, que aceita como parâmetros a lotação do parque e o número máximo de clientes com acesso ao parque. Considere que inicialmente o parque não tem clientes e se encontra vazio. Implemente também os membros-função:

```
unsigned ParqueEstacionamento::getNumLugares() const;
```

```
unsigned ParqueEstacionamento::getNumMaximoClientes() const;
```

Estas funções retornam, respetivamente, a lotação do parque e o número máximo de clientes.

- b) Implemente os membros-função:

```
int ParqueEstacionamento::posicaoCliente(const string & nome) const;
```

```
bool ParqueEstacionamento::adicionaCliente(const string & nome);
```

Estas funções retornam, respetivamente:

- o índice no vetor *clientes* do cliente de nome *nome*, retornando *-1* caso não exista
- o sucesso (*true*) ou insucesso (*false*) na adição/registo de um novo cliente ao parque de estacionamento.

Considere que o cliente está inicialmente fora do parque.

- c) Implemente o membro-função:

```
bool ParqueEstacionamento::entrar(const string & nome);
```

Esta função regista a entrada de um cliente no parque. Retorna *false* se o cliente não puder entrar (não está registado, a sua viatura já está dentro do parque, ou o parque está completo).

- d) Implemente o membro-função:

```
bool ParqueEstacionamento::retiraCliente(const string & nome);
```

Esta função retira o registo do cliente de nome *nome* do parque de estacionamento. A remoção do cliente só é possível se este estiver atualmente fora do parque.

- e) Implemente o membro-função:

```
bool ParqueEstacionamento::sair(const string & nome);
```

Esta função regista a saída de um cliente do parque. Retorna *false* se o cliente não puder sair (não está registado ou a sua viatura não está dentro do parque).

- f) Implemente os membros-função:

```
unsigned ParqueEstacionamento::getNumLugaresOcupados() const;
```

```
unsigned ParqueEstacionamento::getNumClientesAtuais() const;
```

Estas funções retornam o número de lugares ocupados no parque e o número de clientes registados, respetivamente.