1ªaula prática - Classes. Membros constantes. Operadores.

Instruções

- Faça download do ficheiro ***aeda1718_fp01.zip*** da página da disciplina e descomprima-o (contém os ficheiros ***Parque.h***, ***Parque.cpp*** e ***Test.cpp***)
- Abra o eclipse e crie um ***novo projeto C++ do tipo Cute Project*** (*File/New/C++ Project/Cute Project*, usando o compilador *MinGW GCC*)
- No ecran seguinte (*Next*), ative a opção "*Copy Boost headers into Project*"
- Importe para a pasta ***src*** do projeto, os ficheiros: ***Parque.h, Parque.cpp*** **e** ***Test.cpp***
- Compile o projeto.
- Execute o projeto como ***CUTE Test*** (*Run As/CUTE Test*). Se surgir a pergunta de qual compilador usar, escolha ***MinGW gdb***.
- 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.*** Poderá executar o projeto como *CUTE Test* quando quiser saber se a implementação que fez é suficiente para passar no teste correspondente.
- 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 int vagas;
    const unsigned int lotacao;
    vector<InfoCartao> clientes;
    const unsigned int numMaximoClientes;
public:
    ParqueEstacionamento(unsigned int lot, unsigned int 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 int getNumLugares() const;
    unsigned int getNumMaximoClientes() const;
    unsigned int getNumLugaresOcupados() const;
    unsigned int 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 int ParqueEstacionamento::getNumLugares() const;

unsigned int 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 int ParqueEstacionamento::getNumLugaresOcupados() const;

unsigned int ParqueEstacionamento::getNumClientesAtuais() const;

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

g) Implemente o operador:

const ParqueEstacionamento & ParqueEstacionamento::operator += (const ParqueEstacionamento & p2);

que incrementa o parque de um valor *p2*. O incremento de *p2* equivale a adicionar aos clientes deste parque os clientes de *p2*. Se tal excede o número máximo de clientes do parque, só são adicionados os primeiros clientes de *p2* até esse limite máximo. Note que podem existir clientes repetidos.