CI1 Parte prática. Duração: 2h00m

Submeter a resolução da prova num ficheiro zip (incluir os ficheiros *.cpp e *.h). Não necessita incluir o ficheiro Tests.cpp

Pretende-se implementar um sistema de gestão de reservas de um dado hotel. Os quartos (classe **Room**) são identificados por um código de identificação (***code***) e pela largura (***roomWidth***) e comprimento (***roomLength***) do quarto. Existem três tipos de quartos: quartos regulares (**RegularRoom**), quartos familiares (**FamilyRoom**) e quartos de luxo (**LuxuryRoom**).

Todos os quartos têm um piso onde se encontram localizados (***floor***). Os quartos familiares têm ainda um número limite de camas de criança que o quarto pode acomodar (***numChildrenBed***). Já para os quartos de luxo é ainda especificado o tipo de cama disponível (***typeBed***).

A partir da área do quarto é possível calcular o preço do quarto que é calculado à taxa de 4€/m2. No caso dos quartos de luxo, ao valor calculado acresce uma taxa de 15%.

A classe **Hotel** identifica uma dada unidade hoteleira numa determinada localidade (***city***). O hotel também tem uma capacidade máxima (***maxCapacity***) (em número total de quartos que uma dada unidade hoteleira pode ter). O hotel guarda ainda numa lista (***reservationsDone***) os quartos já reservados anteriormente. Esta lista pode conter elementos repetidos.

As classes ***Hotel***, ***Room***, ***RegularRoom***, ***FamilyRoom*** e ***LuxuryRoom*** estão <u>parcialmente</u> definidas a seguir.

<u>NÃO PODE</u> acrescentar membros-dado à classe ***Room***, ***RegularRoom***, ***FamilyRoom*** e ***LuxuryRoom***, para além dos que já se encontram declarados.

```
template <class Chamber>
class Hotel {
   vector<Chamber *> chambers;
   const unsigned int maxCapacity;
   const string city;
   const int numFloors;
   list<Chamber> reservationsDone;
public:
   bool addChamber(Chamber *chamber);
   Chamber* removeChamber(string id,
                          int floor);
   float avgPrice(string municipality) const;
   void sortChambers();
    bool doReservation(string code,int floor);
    list<Chamber> roomsNeverReserved() const;
};
```

```
class Room {
   const string code;
   const int floor;
   float width;
   float length;
   float area;
   bool reservation;
public:
   Room(string b, string m);
   string getCode() const;
   float getWidth() const;
   void setWidth(float roomWidth);
    bool operator == (const Room &room)
                                  const;
   };
```

```
class RegularRoom : public Room {
   public:
   RegularRoom(string code, int floor,
               int roomWidth, int roomLength);
};


class FamilyRoom : public FamilyRoom {
   int numChildrenBed;
public:
   FamilyRoom(string code, int floor,
               int roomWidth, int roomLength,
                  int numChildrenBed);
};
```

```
class LuxuryRoom : public Room {
   string typeBed;
public:
   LuxuryRoom(string code, int floor,
              int roomWidth, int roomLength,
                 string typeBed);
};
```

NOTA: Relembra-se que o uso de iteradores em classes genéricas deve ser precedido da keyword ***typename*** (ex: typename std::vector<Chamber *>::const_iterator it;)

a) [**2.5 valores**] Implemente na classe ***Room*** o operador ==. Dois quartos são iguais se tiverem o mesmo código (***code***) e piso (***floor***).

```
bool operator == (const Room &room) const;
```

b) [**2.5 valores**] Implemente, nas classes necessárias, o membro-função que calcula o preço de um quarto:

```
float price() const
```

Considere, tal como indicado no texto, que o preço é calculado à taxa de 4€/m2. No caso dos quartos de luxo, ao valor calculado acresce uma taxa de 15%.

c) [**2.5 valores**] Implemente na classe genérica ***Hotel*** o membro-função:

```
bool addChamber(Chamber *chamber);
```

Esta função deve adicionar ao vector ***chambers*** o argumento ***chamber***, mas apenas se o hotel não tiver atingido o número máximo de quartos (membro dado **max*Capacity***), e se o quarto a adicionar ainda não existir. Se o quarto for adicionado, a função deve retornar ***true***, senão deve retornar ***false***.

d) [**2.5 valores**] Implemente na classe genérica ***Hotel*** o membro-função:

```
void sortChambers();
```

Esta função ordena os elementos do vector ***chambers***. Note que, no ficheiro de teste, a classe genérica ***Hotel*** é convertida em classe ordinária usando a classe ***Room***. Os quartos devem ficar ordenados por ordem crescente de código (***code***) e, para um mesmo código, por ordem decrescente de piso.

e) [**2.5 valores**] Implemente na classe genérica ***Hotel*** o membro-função:

```
Chamber* removeChamber(std::string code, int floor);
```

Esta função remove do vector ***chambers*** o quarto com o código ***code*** localizado no piso ***floor***, retornando o elemento removido. No caso de o quarto não existir, a função deve lançar a exceção ***NoSuchChamberException*** (já fornecida).

f) [**2.5 valores**] Implemente na classe genérica ***Hotel*** o membro-função:

```
float avgArea(int floor) const;
```

Esta função calcula e retorna a área média dos quartos de um dado piso de uma unidade hoteleira. No caso de não haver quartos nesse piso, a função deve lançar uma exceção ***NoSuchFloorException***, a qual deve conter um membro-dado que é o número do piso e um membro função de acesso a esse membro-dado, chamado ***getFloor***.

g) [**2.5 valores**] Implemente na classe genérica ***Hotel*** o membro-função:

```
bool doReservation(std::string code, int floor);
```

Esta função reserva o quarto com o código ***code*** localizado no piso ***floor***. Se o quarto não existir no vetor ***chambers*** ou já estiver reservado (membro-dado ***reservation*** igual a true), a função retorna false. Senão, coloca o membro-dado ***reservation*** igual a true e adiciona o quarto reservado no final da lista ***reservationsDone***.

h) [**2.5 valores**] Implemente na classe genérica ***Hotel*** o membro-função:

```
list<Chamber> roomsNeverReserved() const;
```

Esta função retorna a lista de quartos do hotel que nunca foram reservados (não estão presentes na lista ***reservationsDone***). Relembre que um quarto é identificado pelo código (membro-dado ***code***) e andar (membro-dado ***floor***)

Submeter a resolução da prova num ficheiro zip (inclui ficheiros *.cpp e *.h). Não necessita incluir o ficheiro Tests.cpp