Parte prática. Duração: 2h00m

Considere uma aplicação para gestão dos correios (CTT). Uma central de correio (classe **Central**) gere a correspondência de múltiplas agências (classe **Agencia**). Uma agência de correios tem um nome e uma lista com a correspondência recebida para expedição. A correspondência (classe **Correspondencia**) pode ser de dois tipos: correio normal (classe **CorreioNormal**); ou EMS (Express Mail Sevice – classe **EMS**).

A correspondência **CorreioNormal** tem um preço fixado na altura da criação de um objecto e é guardado no membro-dado *preco*. A correspondência do tipo **EMS** tem um preço que é 4 vezes o seu peso.

As classes **Central**, **Agencia**, **Correspondencia**, **CorreioNormal** e **EMS** estão parcialmente definidas a seguir.

```
class Central {
   int ID;
   vector<Agencia> agencias;
public:
   Central(int id);
   // …
};

class Agencia {
   string nome;
   vector<Correspondencia *> cartas;
public:
   Agencia(string nome);
   // …
};
```

```
class Correspondencia {
protected:
   string destino;
   int prioridade;
public:
   Correspondencia(...);
   // …
};

class CorreioNormal: public Correspondencia {
   int preco;
public:
   CorreioNormal(...);
   // …
};

class EMS: public Correspondencia {
   int peso;
public:
   EMS(...);
   // …
};
```

a) [**2.5 valores**] Implemente um predicado (função booleana) da classe **Agencia** que verifica se, numa lista de correspondências, estas estão ordenadas por ordem crescente de preço. Implemente o membro-função:

bool estaOrdenado() const

Acrescente todos os membros-função que pensa serem necessários (mesmo em outras classes).

b) [**3 valores**] A central de correios recolhe das agências a correspondência para ser expedida. O processamento é feito por regiões. Implemente na classe Agencia um membro-função que recebe dois nomes de destinos e devolve toda a correspondência com destino entre *destino1* e *destino2* inclusivé. "Entre" dois destinos inclui todos os destinos cujo nome é alfabeticamente igual ou maior que destino1 e inferior ou igual a destino2. Implemente na classe **Agencia** o membro-função:

*vector<Correspondencia *> recolha(string destino1, string destino2);*

Esta função remove do vetor *cartas* da agência a correspondência que satisfaz as condições pedidas e retorna num vetor a correspondência removida. O vetor de correspondências a retornar deve estar ordenado por ordem crescente do destino.

Nota: no vetor da correspondência pode haver mais do que uma correspondência para o mesmo destino.

c) [**3 valores**] Pretende-se localizar uma correspondência para um destino especificado. Para esta tarefa implemente na classe **Agencia** o membro-função:

*Correspondencia *enviarPara(string destino) const;*

Esta função devolve a primeira (a contar do início do vetor) correspondência que se destine a **destino**. No caso de não haver nenhuma correspondência a ser enviada para **destino** deve ser gerada uma exceção do tipo DestinoInexistente. Os objetos deste tipo de exceção guardam o nome do destino em falta e disponibilizam o membro-função de acesso getDestino() para acesso ao nome guardado. Assuma que o membro-dado cartas está ordenado por ordem crescente do valor do membro-dado destino.

d) [**3 valores**] Implemente na classe **Central** os membro-função:

Agencia agenciaPrioritaria(int posicao)

Agencia agenciaMaior(int posicao)

Estas funções retornam respetivamente a agência na posição **posicao**, na ordem de prioridades (*agenciaPrioritaria()*) e na ordem de tamanhos (*agenciaMaior()*). A **posicao** é contada do fim para o princípio quando o vetor está ordenado por ordem crescente. A prioridade de uma agência é a soma das prioridades da sua correspondência, o tamanho é o número de correspondências que possui. Assuma que **posicao** está sempre dentro da gama de índices do vetor *agencias*.

e) [**3 valores**] Implemente uma função global que devolve o nome do destino da primeira correspondência EMS, entre os destinos dest1 e dest2, que pertence à região Norte. Uma vez mais os destinos são considerados maiores/menores de acordo com a comparação alfabética dos seus nomes . Implemente o membro-função:

string naRegiaoNorte(vector<EMS> &cartas, string dest1, string dest2)

Considere que o vetor de correspondências **está** ordenado por ordem crescente dos destinos e que pode haver mais do que uma correspondência EMS para o mesmo destino. A função deve devolver astring vazia (“”) caso não encontre uma correspondência com a prioridade pretendida. A região Norte é composta pelos destinos: “barcelos”, “viana” e “valenca”.

f) [**2.5 valores**] Implemente uma função global *template* “combinaVetores” que recebe dois vetores guardando elementos do mesmo tipo e devolve um vetor com a “combinação” dos dois vectores recebidos. Isto é, o vetor resultado tem todos os elementos dos dois vetores mas não tem repetidos. Os vetores de entrada podem não estar ordenados, mas o vetor devolvido deve estar ordenado por ordem crescente e NÂO pode ter elementos repetidos. Assuma que os vetores originais não têm elementos repetidos.

template<class T> vector<T> combinaVetores(vector<T> vec1, vector<T> vec2)

NOTA: implemente a função no ficheiro Correios.h.

g) [**3 valores**] Implemente a função associada à classe **Central**:

int getAgenciasGeradas()

Esta função devolve o número de objectos do tipo **Agencia** criados desde o início da execução do programa. Faça as alterações que pensa necessárias ao seu código.