Parte prática. Duração: 2h00m

Uma agência bancária pretende implementar um sistema para gestão de informação relativa aos seus clientes e respetivas contas. A informação sobre um cliente (**Cliente**) inclui o nome e ainda o conjunto de contas das quais é 1ºtitular.

A informação sobre uma conta inclui o número de conta, indicação do 2º titular (caso exista), o saldo e o número de transações efetuadas. As transações possíveis de existir são apenas depósitos, levantamentos e custos de manutenção.

Existem dois tipos de conta: normal e de operação. Numa conta normal (**Normal**), os depósitos e levantamentos apenas afetam o saldo e o número de transações. No final do mês é debitada uma taxa de serviço de 1.5 euros, transação relativa a custos de manutenção. Numa conta de operação (**DeOperacao**), cada levantamento de valor superior a 30 euros, gera uma taxa de serviço. Essa taxa é de 0.03 euros para clientes antigos e 0.05 euros para novos clientes. O levantamento e a taxa correspondente contam como uma única transação de levantamento. A transação de depósito não gera nenhuma taxa.

As classes **Agencia, Conta, Normal, DeOperacao, Cliente** e **Gerente** estão parcialmente definidas a seguir.

NÃO PODE acrescentar membros-dado nas classes **Conta**, **Normal** e **DeOperacao.**

```
class Conta {
protected:
  int numConta;
  float saldo;
  int numTransacoes;
  Cliente *titular2;
public:
   Conta(int nConta, float sd=0,
                int nTrans=0);
   // …
};
```

```
class Normal: public Conta {
public:
   Normal(int nConta, float sd=0,
                int nTrans=0);
 // …
};
```

```
class DeOperacao: public Conta {
  float taxa;
public:
  DeOperacao(int nConta, float tx,
           float sd=0, int nTrans=0);
   // …
};
```

```
class Cliente {
  string nome;
  vector<Conta *> contas;
public:
  Cliente(string nm);
  void adicionaConta(Conta *c1);
   // …
};
```

```
class Gerente {
  string nome;
  int ID;
  vector<Cliente *> meusClientes;
public:
  Gerente(string nm);
  //…
};
```

```
class Agencia {
  string designacao;
  vector<Cliente *> clientes;
  vector<Gerente> gerentes;
public:
  Agencia(string desig);
  void adicionaCliente(Cliente cli1);
   // …
};
```

a) [**2.5 valores**] A realização de uma transação de levantamento ou depósito implica uma atualização do saldo da conta e o incremento do membro-dado `numTransacoes`. Implemente, nas classes que considerar necessário, os membros-função responsáveis por estas transações:

bool levantamento(float valor)

void deposito(float valor)

A transação de levantamento só pode ser realizada se o saldo da conta for superior ou igual ao valor a levantar (não considerar o valor da taxa de serviço); nesse caso a função retorna `true`, caso contrário retorna `false`.

b) [**3 valores**] Implemente na classe **Agencia** o membro-função:

*Conta *levantamento(string nomeCli, float valor)*

Esta função efetua o levantamento de `valor` de uma das contas do cliente de nome `nomeCli`. A transação de levantamento é realizada na primeira conta do cliente que possua saldo suficiente, sendo retornado um apontador para essa conta. Se o cliente não existe ou não possui nenhuma conta com saldo suficiente, a função retorna uma `Conta` com número de conta igual a -1.

c) [**2 valores**] Implemente na classe **Agencia** o membro-função:

float fimMes() const

Esta função atualiza todas as contas existentes, debitando a taxa de serviço correspondente às contais normais. Retorna a soma dos saldos de todas as contas da agência, após esta atualização.

d) [**3 valores**] Um cliente pretende sair desta agência bancária. Implemente na classe **Agencia** o membro-função:

vector<Conta> removeCliente(string nomeCli)*

Esta função retira o cliente de nome `nomeCli` do vetor clientes da agência. Desassocia este cliente de todas as contas das quais é 2º titular. As contas das quais é 1ºtitular são passadas para o 2ºtitular, caso exista, senão deixam de existir na agência. A função retorna estas últimas, isto é, as contas das quais o cliente é 1º titular e que não possuem 2ºtitular.

e) [**3 valores**] Implemente na classe **Agencia** o operador <. Uma agência é menor que outra se o saldo médio por cliente é menor. *Nota:* O saldo médio por cliente é igual ao quociente entre a soma dos saldos de todas as contas da agência e o número de clientes.

f) [**2.5 valores**] Implemente na classe **Agencia** o operador de função que aceita um argumento do tipo *string* (representando o nome de um cliente) e retorna o saldo total deste cliente. Deve retornar -1, caso o cliente não exista.

g) [**3 valores**] A agência guarda ainda informação sobre os seus gerentes de conta. Os gerentes são identificados por um ID diferente e sequencial (inteiro). Implemente na classe **Agencia** o membro-função que inicializa esta numeração, de modo a que o próximo gerente tenha ID igual a `IDinicio`:

setGerenteID (int IDinicio)

Implemente ainda na classe **Agencia** o membro-função:

adicionaGerente(string nomeGer)

Esta função adiciona à agência um novo gerente de nome `nomeGer`.