O objetivo desse projeto é praticar os conceitos de Tipos Abstratos de Dados (TAD) e Modularização.

Você foi contratado por um pequeno banco para implementar um sistema de forma a automatizar as suas operações. Basicamente os clientes podem criar contas, depositar, sacar e fazer transferências (pix). Para implementar o seu sistema você vai criar 2 TADs e um arquivo main:

TAD Conta Bancária

O TAD Conta Bancária armazena os dados da conta: id (inteiro), cliente (string) e saldo (float) e possui as seguintes operações:

- Construtor: inicializa uma nova conta com um id e cliente passados como parâmetro e saldo igual a 0.

- Depósito: recebe um valor como parâmetro e adiciona ao saldo da conta.

- Saque: recebe um valor como parâmetro e subtrai do saldo da conta.

- Pix: recebe como parâmetros um apontador para uma outra conta e um valor, e transfere o valor da conta que está chamando o método para a conta passada como parâmetro.

- Imprime: imprime em uma linha o id, nome do cliente e saldo, separados por espaço e com um endl no final. O saldo deve ser impresso com 2 casas decimais.

TAD Banco:

O TAD banco é usado para armazenar as contas bancárias. Basicamente, ele vai ter como atributos um número inteiro que armazena o número total de contas e um vetor com apontadores para contas. (Para simplificar, você pode considerar que o número máximo de contas do Banco é 20 e usar um vetor simples com alocação estática). O TAD Banco possui as seguintes operações:

- Construtor: inicializa o TAD colocando o valor 0 no número de contas e inicializando todos elementos do vetor de contas com nullptr.

- CriaConta: cria uma nova ContaBancária com id e nome passados como parâmetro e a coloca no vetor de contas, incrementando também o número de contas. Note que não podem haver contas com IDs repetidos, portanto o seu método deve retornar um apontador para a conta criada ou, em caso de falhas, nullptr.

- Pesquisa: recebe um id de uma conta e faz uma pesquisa por uma conta com esse id no vetor de contas, retornando um apontador para a conta encontrada ou nullptr caso a conta não exista.

- ListaContas: imprime as informações de todas as contas cadastradas no Banco (id, cliente, saldo) uma conta por linha.

Main:

O seu programa principal faz a gerência das operações do banco. Ele vai ser basicamente um loop que lê comandos da entrada (C, D, S, P, I, T) juntamente com os parâmetros e faz as operações de acordo. Para cada comando, o sistema imprime uma mensagem com uma quebra de linha (endln) no final. Importante: as mensagens devem ser exatamente como especificado abaixo para evitar erros na correção automática. As operações são:

C <id> <nome>: cria uma conta com id e nome indicados. Deve ser impresso: “conta criada” em caso de sucesso ou “ERRO: conta repetida” em caso de falha.

D <id> <valor>: faz um depósito do valor na conta indicada. Você poderá considerar que os valores fornecidos serão sempre positivos. Deve ser impresso: “depósito efetuado” em caso de sucesso ou “ERRO: conta inexistente” caso o id da conta seja inválido.

S <id> <valor>: faz um saque do valor na conta indicada. Você poderá considerar que os valores fornecidos serão sempre positivos. Deve ser impresso: “saque efetuado” em caso de sucesso ou “ERRO: conta inexistente” caso o id da conta seja inválido.

P <id> <dest> <valor>: faz um pix, transferindo o valor da conta indicada por <id> para a conta indicada por <dest>. Você poderá considerar que os valores fornecidos serão sempre positivos. Deve ser impresso: “pix efetuado” em caso de sucesso ou “ERRO: conta inexistente” caso as uma das contas indicadas sejam inválidas.

I: lista as informações de todas as contas cadastradas no Banco (id, cliente, saldo) uma conta por linha.

T: termina a execução do programa.

Segue abaixo um exemplo de uma execução do Sistema:

C 123 Luiz
conta criada
C 123 Maria
ERRO: conta repetida
C 999 Maria
conta criada
D 123 1000
deposito efetuado
S 555 500
ERRO: conta inexistente
S 123 200
saque efetuado
P 123 999 300
pix efetuado
I
123 Luiz 500.00
999 Maria 300.00
T

Você deverá submeter 5 arquivos: ContaBancaria.hpp, ContaBancaria.cpp, Banco.hpp, Banco.cpp e main.cpp. No seu programa principal, você não deverá acessar diretamente os atributos dos seus TADs. Todo o acesso deverá ser feito através dos métodos implementados.