Quarto Exercício-Programa

Prof. Luciano Antonio Digiampietri

Prazo máximo para a entrega: 23/07/2021

1 Sistema de Gerenciamento de um Banco

Neste trabalho, você deverá desenvolver um sistema simplificado para o gerenciamento de um banco. De uma maneira resumida (que será detalhada ao longo deste enunciado), um banco é composto por gerentes, os quais gerenciam um conjunto de clientes (que podem ou não ser clientes especiais). Há apenas duas características dos clientes a serem gerenciadas: o valor em suas contas correntes e o valor de suas dívidas/empréstimos com o banco. Um cliente pode estar associado a mais de um gerente. Cada banco pode ter até 10 gerentes e cada gerente pode gerenciar até 20 clientes.

Detalhamento:

Nesta simplificação existem as seguintes estruturas principais, descritas a seguir: *PESSOA*, *GERENTE*, *CLIENTE* e *BANCO*.

***PESSOA*:** é uma estrutura que possui dois campos (*nome* do tipo ponteiro para caracteres e *cpf* do tipo inteiro).

***CLIENTE*:** possui um campo chamado *pessoa* do tipo PESSOA e três campos adicionais: um campo do tipo char chamado de *tipo* cujo valor será 'C' para cliente regular ou 'E' para cliente especial e duas variáveis do tipo inteiro *valorContaCorrente* que armazena o valor que o cliente possui em sua conta corrente e *valorDaDivida* que armazena o valor que o cliente está devendo ao banco.

***GERENTE*:** possui um campo chamado *pessoa* do tipo PESSOA e dois campos adicionais: um arranjo de referências/ponteiros para CLIENTE, chamado *clientes* e um inteiro chamado *numClientes* que indica quantos clientes há dentro do arranjo *clientes*.

***BANCO*:** possui dois campos: um arranjo de referências/ponteiros para GERENTE, chamado *gerentes* e um inteiro chamado *numGerentes* que indica quantos gerentes o banco possui (estes gerentes estarão no arranjo *gerentes*).

O código fornecido para este EP já possui várias funções implementadas, conforme será detalhado adiante, e possui uma função ***main*** que, quando executada, cria um cenário de um banco com gerentes e clientes e testa diversas funções envolvidos neste EP. Esta função não fará parte da avaliação do EP e serve apenas para te auxiliar nos testes de seu EP (ela, não necessariamente, cobre todos os testes possíveis para este EP).

O EP contém duas constantes relacionadas ao Sistema de Gerenciamento do Banco: *dividaMaxima* e *dividaMaximaEspecial* que correspondem ao valor máximo de dívida/empréstimo que um cliente ou cliente especial, respectivamente, podem ter valor igual a 30000 ou 50000.

Já foram implementadas funções para criar/inicializar cada uma das "entidades" relacionadas ao sistema: *novoCliente*, *novoClienteEspecial*, *novoGerente* e *novoBanco*.

Há, também, duas funções já implementadas para exibir dados relacionados ao sistema, são elas: *void imprimirClientes(GERENTE ger1)*, que exibe algumas informações dos clientes gerenciados pelo gerente *ger1* (ver código fornecido para mais informações); e *void imprimirDadosBanco(BANCO b1)*, que exibe algumas informações do banco *b1*, de seus gerentes e dos clientes desses gerentes (ver código fornecido para mais informações).

As seguintes funções são relacionadas aos gerentes (todas já estão implementadas).

- *bool adicionarCliente(GERENTE* g1, CLIENTE* cliente)*: função para adicionar um cliente (ou mais especificamente uma referência a um cliente) no arranjo de clientes do gerente referenciado por *g1*. Caso o número de clientes seja igual a 20, não deve adicionar e deve retornar *false*. Caso contrário, há duas situações: 1ª: o cliente já consta no arranjo de referências a clientes (verifique isso usando o número do CPF), neste caso o cliente não deve ser reinserido e a função deve retornar *false*; 2ª: o cliente passado como parâmetro não consta no arranjo de clientes, neste caso o cliente deve ser adicionado na posição *numClientes*, este campo (apontado por g1) deve ser incrementado em 1 e a função deve retornar *true*.
- *void cobrarTodosEmprestimos(GERENTE g1)*: função para cobrar os empréstimos de todos os clientes do gerente *g1*. Para cada um dos clientes referenciados no arranjo *clientes* do gerente *g1*, esta função deve: não fazer nada para o cliente, caso seu *valorDaDivida* seja igual a zero; caso contrário, há duas situações: 1ª: se *valorContaCorrente* do cliente for maior ou igual ao *valorDaDivida*, deve fazer o cliente pagar a dívida, isto é, o *valorContaCorrente* será atualizado, descontando-se o valor da dívida e o *valorDaDivida* será zerado. 2ª: se o *valorContaCorrente* do cliente for menor do que o *valorDaDivida*, deve fazer o cliente pagar parte da dívida, isto é, o *valorDaDivida* será atualizado, tendo seu valor diminuído pelo *valorContaCorrente* e o *valorContaCorrente* será zerado.

Há quatro funções relacionadas aos clientes (que deverão ser implementadas/completadas por você):

- *bool negativado(CLIENTE* c1)* : função que retorna *true* caso, para o cliente referenciado por *c1*, *valorContaCorrente* seja menor do que *valorDaDivida*. Caso contrário, deverá retornar *false*.
- *bool obterEmprestimo(CLIENTE* c1, int valor)*: função para o cliente atual (referenciado por *c1*) obter um empréstimo de acordo com o valor passado por parâmetro. Caso

o valor do parâmetro *valor* seja menor ou igual a zero, a função deve retornar *false*. Caso contrário há duas situações: 1ª: se o valor do parâmetro mais o valor do campo *valorDaDivida* seja maior do que o valor da constante *dividaMaxima* para clientes do tipo 'C' ou *dividaMaximaEspecial* para cliente especiais (isto é, tipo igual a 'E'), a função deve retornar *false*; 2ª: caso contrário, o campo *valorDaDivida* deve ser incrementado em *valor*, o campo *valorContaCorrente* deve ser incrementado em *valor* e a função deve retornar *true*.

- *bool pagarEmprestimo(CLIENTE* c1, int valor)*: função para o cliente atual (referenciado por *c1*) pagar parte de sua dívida de acordo com o valor passado por parâmetro. Caso o valor do parâmetro seja menor ou igual a zero, a função deve retornar *false*. Caso contrário, há duas situações: 1ª: se o valor do parâmetro for maior do que o *valorDaDivida* ou for maior do que *valorContaCorrente*, a função deve retornar *false*; 2ª: caso contrário, o campo *valorDaDivida* deve ser decrementado em *valor*, o campo *valorContaCorrente* deve ser decrementado em *valor* e a função deve retornar *true*.

- *bool realizarSaque(CLIENTE* c1, int valor)*: função para o cliente atual (referenciado por *c1*) realizar um saque do valor passado por parâmetro. Caso o valor do parâmetro seja menor ou igual a zero, a função deve retornar *false*. Caso contrário há duas situações: 1ª: se o valor do parâmetro for maior do que o valor do campo *valorContaCorrente*, a função deve retornar *false*; 2ª: caso contrário, o campo *valorContaCorrente* deve ser decrementado em *valor* e a função deve retornar *true*.

Além da função que exibe informações relacionadas a um banco, há uma outra função relacionada ao banco, a qual deverá ser implementada por você. Note que essa função tem um comportamento muito parecido com outra função já implementada no EP, chamada *adicionarCliente*.

bool adicionarGerente(BANCO b1, GERENTE* gerente)*: função para adicionar um gerente (uma referência a um gerente) no arranjo de gerentes do banco referenciado por *b1*. Caso o número de gerentes seja igual a 10, não deve adicionar e deve retornar *false*. Caso contrário, há duas situações: 1ª: o gerente já consta (já é referenciado) no arranjo de gerentes (verifique isso usando o número do *CPF*), neste caso o gerente não deve ser reinserido e a função deve retornar *false*; 2ª: o gerente passado como parâmetro não consta no arranjo de gerentes: o gerente deve ser adicionado na posição *numGerentes*, este campo deve ser incrementado em 1 e a função deve retornar *true*.

1.1 Material a Ser Entregue

Um arquivo, denominado `NUSP.c` (sendo NUSP o seu número USP, por exemplo: 123456789.c), contendo seu código, incluindo todas as funções solicitadas e qualquer outra função adicional que ache necessário. Para sua conveniência, `completeERenomeie.c` será fornecido, cabendo a você então completá-lo e renomeá-lo para a submissão.

Atenção!

1. Não modifique as assinaturas das funções já implementadas e/ou que você deverá completar!
2. Para avaliação, as diferentes funções serão invocadas diretamente (individualmente ou em conjunto com outras funções). Em especial, qualquer código dentro da função `main()` será ignorado.

2 Entrega

A entrega será feita única e exclusivamente via sistema e-Disciplinas, até a data final marcada. Deverá ser postado no sistema um arquivo c, tendo como nome seu número USP:

```
seuNumeroUSP.c   (por exemplo, 12345678.c)
```

Não esqueça de preencher o cabeçalho constante do arquivo .c, com seu nome, número USP e turma etc.

A responsabilidade de postagem é exclusivamente sua. Por isso, submeta e certifique-se de que o arquivo submetido é o correto (fazendo seu download, por exemplo). Problemas referentes ao uso do sistema devem ser resolvidos com antecedência.

3 Avaliação

A nota atribuída ao EP terá como foco principal a funcionalidade solicitada, porém não esqueça de se atentar aos seguintes aspectos:

1. Documentação: se há comentários explicando o que se faz nos passos mais importantes e para que serve o programa (Tanto as funções quanto o programa em que estão inseridas);
2. Apresentação visual: se o código está legível, identado etc;
3. Corretude: se o programa funciona.

Além disso, algumas observações pertinentes ao trabalho, que influem em sua nota, são:

- Este exercício-programa deve ser elaborado individualmente;
- Não será tolerado plágio, em hipótese alguma;
- Exercícios com erro de sintaxe (ou seja, erros de compilação), receberão nota ZERO.