

Trabalho 02
Versão: 3 de julho de 2018

1 Instruções:

- O trabalho é em equipe de no máximo **3 alunos**.
- O nome dos integrantes devem estar no cabeçalho comentado em todos os arquivos.
- Será aplicado detector de plágio! Uma vez detectado plágio, a **nota da disciplina** será **zerada**.
- Data de entrega: **29 de julho** no sistema do moodle até as 23:59.
- A entrega deve ser no formato ZIP com o seguinte conteúdo:
 - + Código-fonte (bem como os arquivos `.c` e `.h` referentes aos TAD utilizados);
 - + Arquivo Makefile¹ associado para compilar o código.
- O código deve estar escrito em **Linguagem C** e as únicas bibliotecas permitidas são: `stdio.h`, `stdlib.h` e `string.h` caso contrário o trabalho será invalidado.
- Para validação do código será utilizado um ambiente GNU/Linux com o compilador **GCC** (versão 7+) dado os seguintes parâmetros²:
`gcc -Wall -Wextra -Werror -Wpedantic`
- Para verificação de vazamento de memória e gerência dos ponteiros, será utilizado o software **Valgrind**³.

¹Veja mais sobre em https://pt.wikibooks.org/wiki/Programar_em_C/Makefiles.

²Leia mais sobre em: <https://gcc.gnu.org/onlinedocs/gcc-6.4.0/gcc/Warning-Options.html>.

³<http://valgrind.org>

Otimização de operações bancárias – Um protótipo

Um banco deseja melhorar seu sistema de operações bancárias. Atualmente o sistema deles armazena todas as informações dos clientes e uma lista ordenada. A proposta do banco é para sua equipe desenvolver uma organização dos dados em forma de Árvore AVL, pois o gerente de T.I. do banco sabe que as operações em uma AVL têm custo de $O(\log n)$. Desta forma, você estará focado em elaborar um TAD para uso de Árvore AVL e uma interface de teste que seja capaz de receber dados pela entrada padrão e computar a saída.

Simulação

A entrada constará de uma sequência de operações que seu programa deve conseguir computá-las e retornar uma saída determinada. Assim como na versão completa da simulação, será informada uma quantidade N de operações. Cada operação será identificada com uma letra seguida dos seus devidos parâmetros. Os dados de entrada são as informações dos cliente: Código_cliente: Inteiro; Operação: 1 para saque e 0 para depósito; e Valor: valor inteiro. A **chave de ordenação** deve ser o código do cliente.

A árvore deve guardar as informações e gerar um relatório descrito posteriormente que consta na quantidade de clientes distintos que foram inseridos bem como as informações referentes ao código do cliente, quantidade de operações e saldo final.

2 Formato de entrada

A entrada constará de uma sequência de operações, terminada com uma **operação de término de sequência de operações**. As operações que podem ocorrer são:

- **insere nó:** esta operação consiste de uma linha contendo a letra 'i', seguida de um espaço e três números inteiros separados com um espaço que serão respectivamente código_cliente, operação e valor. Esta operação causa a inserção de um nó na árvore cuja chave será o número inteiro referente ao código_cliente. Caso já haja nó na árvore com valor de chave igual a este número, o nó com tal chave deve ter seus dados atualizados de acordo com a operação informada.
- **consulta nó:** esta operação consiste de uma linha contendo a letra 'c', seguida por um espaço, seguido por um valor inteiro. Esta operação verifica se há ou não nó na árvore com valor de chave igual ao número inteiro digitado.
- **remove nó:** esta operação consiste de uma linha contendo a letra 'r', seguida por um espaço, seguido por um valor inteiro. Esta operação retira o nó da árvore que tiver chave igual ao número inteiro indicado, se houver. Se não houver, esta operação não gera efeito.
- **lista chaves dos nós da árvore em ordem:** esta operação consiste de uma linha contendo a letra 'p', seguida de um espaço, seguido da letra 'c' ou 'd'. Se a segunda letra é 'c', esta operação lista as chaves dos nós da árvore em ordem crescente. Se a segunda letra for 'd', esta operação lista as chaves dos nós da árvore em ordem decrescente.
- **lista chaves de um determinado nível da árvore:** esta operação consiste de uma linha contendo a letra 'n', seguida de um espaço, seguido de um número inteiro maior ou igual a 1. Esta operação lista as chaves dos nós da árvore que tiverem nível igual ao número fornecido. Assuma que a raiz da árvore possui nível 1.

- **informa altura:** esta operação consiste de uma linha contendo a letra ‘h’. Esta operação informa a altura da árvore.
- **término da entrada:** a sequência de operações será terminada por uma linha com a letra ‘f’.

3 Formato da Saída

Cada operação deverá apresentar a saída da seguinte forma:

- **insere nó:** esta operação não gera saída.
- **consulta nó:** se houver um nó na árvore com a chave informada, esta operação gera, na saída, a sequência de caracteres “**existe no com chave: X**”, onde X é a chave informada. Caso não haja, esta operação gera, na saída, a sequência de caracteres “**nao existe no com chave: X**”, onde X é a chave informada.

Observação: Não usamos nenhum tipo de acentuação.

- **remove chave:** esta operação não gera saída (independentemente se há ou não nó com o valor de chave indicado na operação).
- **lista chaves dos nós da árvore em ordem:** esta operação lista os dados contidos nos nós, um em cada linha. Os dados serão três inteiros que correspondem respectivamente ao código_cliente, quantidade_de_operação e saldo.
- **lista chaves de um determinado nível da árvore:** esta operação lista as chaves dos nós, uma em cada linha. A sequência das chaves dos nós listados deve seguir a ordem da representação gráfica da árvore, da esquerda para a direita.
- **informa altura:** esta operação deve retornar um inteiro que informa a altura da árvore corrente. Considere que a árvore vazia tem altura zero.
- **término da entrada:** esta operação irá gerar um relatório com todos os dados armazenados na árvore. Para tal, deve iniciar imprimindo a seguinte sequência de caractere “**+- Início relatorio +-**”, depois imprimir o inteiro L que representa a quantidade distintas de clientes seguido de L linhas que devem conter três inteiros C_i , Q_i e S_i , para $1 \leq i \leq L$, separados por um espaço em branco e finalizar com a seguinte sequência de caractere “**+- Fim relatorio +-**”. A ordem de impressão dos dados é dada pela operação de remoção da raiz da árvore.

Restrições

Algumas restrições para a implementação das estruturas de dados do sistema:

- Deve-se utilizar a entrada padrão para receber os dados e a saída padrão para imprimir os dados;
- Cada nó da árvore deve, no mínimo, ter três ponteiros onde se referenciam, respectivamente, ao seu filho esquerdo, filho direito e pai;
- O sistema tem que utilizar **fortemente** o conceito de tipo abstrato de dados (TAD) na implementação da árvore AVL, i.e., as definições dos nós e das estruturas de dados da árvore devem estar definidas no arquivo `avl.c` e as interfaces/assinaturas das funções e tipos no arquivo `avl.h`;

- Não deve utilizado em **nenhuma hipótese** variáveis globais tanto no arquivo principal quanto no TAD;
- O conjunto: TAD + arquivo principal deve gerenciar a memória de maneira ótima, i.e., tudo que for alocado deve ser desalocado⁴;
- No arquivo principal, cada operação de entrada possível deve ter uma função para chamar as funções do TAD e formatar sua saída.

Observações

- As operações de remoção, inserção e percurso devem estar de acordo com o que foi visto em sala, pois caso contrário a sua saída será diferente e terá nota prejudicada.

⁴Será utilizado o software *Valgrind* para verificação de corretude.