Universidade Federal de Lavras
Departamento de Ciência da Computação
1º semestre de 2014

GCC109 – Algoritmos e Estrutura de Dados III
(Turmas 10A/14A/22A)

1ª Trabalho prático

Controle de voos e passageiros com árvores AVL

Requisitos

Este trabalho consiste em implementar um sistema para armazenar e buscar informações sobre voos e passageiros com reservas em voos. O sistema seria utilizado por operadores de aeroportos para controlar voos e que passageiros estão neles. A busca por passageiros em um voo em diversos momentos, bem como a busca por voos são operações críticas, e portanto, devem ser realizadas com a maior eficiência. O sistema deverá efetuar a implementação de um TAD para árvore AVL genérico para ser utilizado no sistema de duas formas:

1) O sistema deverá conter um conjunto de **voos** que são realizados em determinados dias. Para otimizar a busca por cada voo, a estrutura de armazenamento de voos deverá ser construída como uma árvore balanceada do tipo AVL. A chave a ser utilizada deverá ser uma combinação do código do voo (com duas letras e quatro números) com a data, separados por hífen (ex. BA0087-23042014). Além da chave, cada nó representando um voo deverá armazenar:
   - Aeroporto de partida (armazenado como um código de três letras)
   - Aeroporto de chegada (no mesmo formato do de partida)
   - Horário do voo (armazenado no formato hh:mm)
   - Companhia aérea (código de duas letras)
2) Para cada voo, serão listados os **passageiros** com reservas para aquele voo. Cada nó representando um voo na árvore de voos deverá conter uma outra árvore AVL que irá conter os passageiros relativos a aquele voo. A chave de cada nó que armazena passageiros deverá ser o número do passaporte (armazenado como uma string de 10 caracteres). Além disso deverá ser armazenado para cada passageiro:
   - Nome completo (até 100 caracteres)
   - Poltrona (3 caracteres).

O trabalho deverá ser implementado de forma que a TAD Árvore AVL seja um módulo separado do programa principal, e deverá conter as seguintes funções. Outras funções podem (e devem!) ser implementadas para efetuar operações necessárias para as funções básicas.

1. Criar árvore vazia
2. Inserir nó
3. Remover nó
4. Buscar nó
5. Imprimir lista de chaves com percurso pré-ordem
6. Imprimir lista de chaves com percurso em-ordem
7. Destruir a árvore

Sugestão: para facilitar a implementação da TAD AVL de forma que aceite tanto nós com registros de voos quanto nós com registros de passageiros, os grupos podem utilizar estruturas com ponteiros para **void** ou estruturas do tipo **union**.

Comandos Permitidos

Os **testes** para avaliação dos trabalhos serão feitos de **forma automática**. Desta forma, é imprescindível que as **entradas e saídas de dados** sejam feitas **exatamente da forma especificada**, incluindo espaços, quebras de linha e caracteres especificados.

Os comandos permitidos são representados por duas letras maiúsculas. Toda linha de entrada obrigatoriamente inicia com um comando. Só serão fornecidos como entrada comandos aqui especificados. Os únicos comandos permitidos como entrada são apresentados a seguir (mas não é necessário tratar commandos inválidos). O símbolo ¶ denota um único espaço em branco.

- **Insere voos**

IV ¶ *codigo_voo-data* ¶ *aeroporto_partida* ¶ *aeroporto_chegada* ¶ *hh:mm* ¶ *companhia_aerea*

Insere um voo na árvore de voos. Se a entrada for feita com sucesso, imprime SUCESSO na tela, e caso contrário imprime ERRO. Só deve ser mantida uma árvore de voos por vez na memória.

- **Buscar voos**

BV ¶ *codigo_voo-data*

Efetua a busca de um voo dentro da árvore, e imprime as demais informações sobre o voo na tela na ordem em que estão armazenadas separadas por espaço. Se o voo não existir, imprimir ERRO.

- **Remove voo**

R ¶ *codigo_voo-data*

Remove um voo na árvore de voos. Se a remoção for feita com sucesso, imprime SUCESSO na tela, e caso contrário imprime ERRO. Ao remover um voo, deve ser destruída a árvore de passageiros contida nele.

- **Imprimir voos**

PV ¶

O comando PP deverá imprimir o conteúdo da árvore contida na memória, imprimindo somente a chave contida em cada nó em uma linha separada. A impressão deverá ser feita utilizando um percurso **pré-ordem**. Cada nó deverá ser impresso em uma linha separada na tela, contendo:

<chave> <fator de balanceamento>

- **Insere passageiros**

IP ¶ *codigo_voo-data* ¶ *passaporte* ¶ *nome* ¶ *poltrona*

Insere um passageiro na árvore de passageiros de um dado voo. Se a entrada for feita com sucesso, imprime SUCESSO na tela, e caso contrário imprime ERRO.

- **Buscar passageiros**

BP ¶ *codigo_voo-data* ¶ *passaporte*

Efetua a busca de um passageiro dentro da árvore de um dado voo, e imprime as demais informações sobre o passageiro na tela na ordem em que estão armazenadas separadas por espaço. Se o passageiro ou o voo não existir, imprimir ERRO.

- **Remove passageiro**

RP ¶ *codigo_voo-data* ¶ *passaporte*

Remove um passageiro da árvore de passageiros de um voo. Se a remoção for feita com sucesso, imprime SUCESSO na tela, e caso contrário imprime ERRO.

- **Imprimir passageiros**

PP ¶ *codigo_voo-data*

O comando PP deverá imprimir o conteúdo da árvore contida na memória para um dado voo, imprimindo somente a chave contida em cada nó em uma linha separada. A impressão deverá ser feita utilizando um percurso **pré-ordem**. Cada nó deverá ser impresso em uma linha separada na tela, contendo:

<chave> <fator de balanceamento>

- **Finalizar programa**

FM

Finaliza o programa, destruindo a árvore e desalocando todos os espaços da memória.

Compilação e documentação

Todos os trabalhos deverão ser implementados utilizando linguagem C padrão ANSI. Todos os trabalhos deverão ser compilados utilizando o compilador GCC (GNU C Compiler) em modo ANSI (com linha de comando 'gcc –ansi'). Também poderá ser utilizada a estrutura de projetos da IDE Code:Blocks, desde que seja verificado que o código é compilado adequadamente no compilador GCC.

Deverão ser fornecidas todas as instruções de compilação e arquivos **makefile**. Deverá ser indicado qual sistema operacional foi utilizado.

Deverá ser efetuada documentação interna de todas as funções implementadas e de partes importantes do código. Também deverá ser gerada uma documentação interna com descrição geral do programa, descrição das funções implementadas (incluindo parâmetros e retorno), da interface do programa cliente e casos de teste realizados.

Caso o grupo utilize a implementação do TAD árvore AVL baseado em algum livro ou outro material, isso deverá ser claramente explicitado na documentação, para que o trabalho não seja classificado como cópia.

Critérios de correção

A **nota do trabalho** terá os seguintes critérios de avaliação:
1. NC é a nota do critério Correção;
2. DO é a nota do critério Documentação;
3. NI é a nota do critério Interface;
4. NCF é a nota do critério Código Fonte.

1. **Correção (NC):** o programa faz o que foi solicitado? Faz tudo o que foi solicitado? Utiliza encapsulamento de informação (i.e., acessa adequadamente os TADs definidos?).
2. **Documentação (DO):** a documentação interna e externa apresenta as principais características do programa e estruturas utilizadas? Documenta as funções, interface e casos de teste? Apresenta instruções de compilação e forma como o programa foi modularizado?
3. **Interface (NI):** utiliza rigorosamente os padrões para entrada e saída de dados especificados?
4. **Código fonte (NCF):** é claro (identação, espaçamento, organização em geral), nomes de variáveis são sugestivos, está bem documentado?

Grupos e entrega

O trabalho poderá ser realizado por grupos de **até 2 alunos** da mesma turma. No entanto, todos os membros do grupo devem participar ativamente da implementação do trabalho, e devem ser capazes de explicar quaisquer aspectos do trabalho. Poderão ser realizadas argüições individuais sobre o trabalho pelo professor.

A entrega deverá ser realizada pelo Moodle em http://alunos.dcc.ufla.br até a meia noite do **dia 18 de maio de 2014**. Só é necessário que um dos membros do grupo faça a entrega. Os nomes dos membros do grupo deverão estar claramente identificados na documentação interna e externa do código. Todos os arquivos contendo os códigos e documentação deverão ser entregues como um único arquivo compactado utilizando .zip ou .tar.gz.

Trabalhos copiados receberão nota zero.

A cada dia de atraso, será descontado 10% da nota do trabalho, até o limite de 4 dias. Trabalhos entregues após 4 dias depois do prazo receberão nota zero.