SCC502 – Algoritmos e Estruturas de Dados I

Prof.ª Maria Cristina Ferreira de Oliveira

2º sem. 2014

Trabalho 4 – Árvores Binárias

Prazo de entrega: 10/12/2014 no Run.Codes

(Trabalho Individual)

Introdução

Você é o responsável pela implementação do sistema de RH de uma empresa multinacional. Serão feitas muitas consultas ao sistema (por Nome e CPF) e deseja-se que todas as consultas sejam realizadas de forma rápida. Faça um programa que permita o **cadastro**, a **remoção** e a **consulta** de funcionários.

Utilize estruturas de dados do tipo Árvore Binária de Busca para armazenar e organizar os dados dos funcionários. Serão necessárias duas árvores: uma ordenada por Nome e uma por CPF. Entretanto, para evitar redundância de dados (informações iguais armazenadas em locais diferentes), o cadastro deverá ser feito em uma estrutura separada que será referenciada pelas árvores (*info). Utilize as estruturas abaixo:

```
typedef struct{
      int CPF;
      char Nome[50];
      char Profissao[30];
}Info;

typedef struct t_no{
      Info *info;
      struct t_no *esq;
      struct t_no *dir;
}No;

No *arvoreCPF;
No *arvoreNome;
```

Observação: Não serão cadastrados dois nomes ou CPFs iguais.

Ao final do programa, libere todos os endereços de memória alocados.

Entrada e Saída

A entrada consiste em várias linhas, cada uma correspondente a um comando no sistema.

1) Inserir:

i <nome_sem_espacos> <CPF> <profissão_sem_espacos>

Insere o funcionário nas três árvores. Como as strings não contém espaços, pode-se ler cada uma com um scanf("%s",chr);

2) Remover por nome:

r n <nome_sem_espacos>

Remove o funcionário das duas árvores e exclui a informação de Info.

3) Remover por CPF:

r c <CPF>

Remove o funcionário das duas árvores e exclui a informação de Info.

4) Buscar por nome:

b n <Nome>

Busca o funcionário na árvore ordenada por nome e imprime:

<Nome> <CPF> <Profissao><\n>, se funcionário encontrado.

0<\n>, se funcionário não encontrado.

5) Buscar por CPF:

b c <CPF>

Busca o funcionário na árvore ordenada por CPF e imprime:

<Nome> <CPF> <Profissao><\n>, se funcionário encontrado.

0<\n>, se funcionário não encontrado.

6) Sair do programa e desalocar endereços de memória:

s

Exemplo de execução:

ENTRADA:

```
i jorge 1234 programador
i ana 3311 analista
i laura 133 gerente
i pedro 1 estagiario
b n pedro
b c 1234
r n jorge
r c 3311
b c 1234
b n ana
b c 133
s
```

SAÍDA

```
pedro 1 estagiario
jorge 1234 programador
0
0
laura 133 gerente
```

Outras Informações Importantes

- O trabalho deve ser feito individualmente.
- O programa pode ser feito na linguagem C.
- Todas as submissões são checadas para evitar cópia/plágio/etc.
- Comente o seu código com uma explicação rápida do que cada função, método ou trecho importante de código faz (ou deveria fazer). Os comentários e a modularização do código (“.c”, “.h”, “Makefile”) serão checados e valem nota.
- Entradas/saídas devem ser lidas/escritas a partir dos dispositivos padrão, ou seja, use as funções “*printf(...)”* e “*scanf(...)”*. Para testar, arquivos podem ser redirecionados para/de seu programa na linha de comando utilizando os operadores < e >.
- Exemplo:

```
# ./trab3 < entrada.txt > saida.txt
```