Universidade Federal de Itajubá
Instituto de Matemática e Computação
Algoritmos e Estruturas de Dados II
CTCO-01
Vanessa Souza

Atividade Avaliativa

Assunto : TAD Árvore Binária de Pesquisa

Implementação de um dicionário de palavras com sugestões de correção automática

Objetivo Pedagógico
Essa atividade oferece prática em manipulação de estruturas de dados, com foco em árvores binárias de busca. Além disso, permite que os alunos compreendam o funcionamento de um dicionário de dados, explorem algoritmos de busca e pratiquem conceitos de gerenciamento de memória em C.

Descrição Geral
Você foi contratado por uma empresa de software para desenvolver uma ferramenta de correção de texto que auxilia escritores e estudantes. Sua tarefa é implementar um dicionário de palavras em C, utilizando uma árvore de pesquisa binária não balanceada. Esse dicionário permitirá a busca e o armazenamento de palavras e seus significados. Além disso, o sistema oferecerá sugestões de correção para palavras que não estão no dicionário, com base no prefixo da palavra digitada pelo usuário.

Objetivo
O objetivo é criar um programa em C que:

1. Armazene um dicionário de palavras e definições usando uma árvore binária de busca.
2. Permita que o usuário:
    - Busque uma palavra para ver seu significado.
    - Adicione novas palavras e definições ao dicionário.
    - Receba sugestões automáticas com base nos prefixos inseridos, caso a palavra exata não seja encontrada (simulando uma função de "autocomplete" básica).

Requisitos do Programa

1. Carregamento Inicial:
    - O programa deve ler um arquivo de palavras e significados. O formato do arquivo é como o exemplificado em entrada.txt. A primeira linha é a palavra. A segunda é o significado, e assim sucessivamente até o final do arquivo.
    - As palavras devem ser armazenadas na árvore binária de busca, onde cada nó representa uma palavra e sua definição.
    - A palavra ter ter, no máximo, 10 caracteres. A definição deve ter no máximo, 40 caracterers.
2. Busca de Palavras e Exibição de Definições:

- O usuário pode buscar uma palavra específica. Se a palavra existir no dicionário, o programa exibe sua definição.
- Caso a palavra não seja encontrada, o programa deve sugerir palavras que começam com o mesmo prefixo, com base na estrutura da árvore (busca por prefixo). O prefixo é definido pelas três primeiras letras da palavras de entrada.

3. Remoção de Palavras:
   - O usuário pode remover uma palavra do dicionário, caso deseje. A estrutura da árvore deve ser ajustada para manter a organização e a ordenação das palavras.
4. Exibição Ordenada:
   - O programa deve ter uma opção que exiba todas as palavras do dicionário em ordem alfabética (ou seja, em ordem crescente das chaves da árvore).
   - O programa deve ter uma opção que exiba todas as palavras do dicionário em pré-ordem.

Regras e Restrições

- Use uma árvore binária de busca não balanceada para implementar o dicionário.
- A árvore deve usar strings como chaves (palavras) e armazenar uma definição (string) como valor.
- As operações de inserção, busca, e exclusão devem ser implementadas manualmente pelos alunos, sem o uso de bibliotecas prontas de árvores.

Exemplo de Funcionamento

Main e saída do programa para quando a palavra é encontrada no dicionário.

```
int main() {
    char nomeArquivo[20] = "entrada.txt";
    char palavra[11] = "Bola";
    dic *D = criaDic();
    carregaDicionario( dicionario: D, nomeArquivo);
    printf("\nPercorrimento em pré-ordem\n");
    percorrePreOrdem( palavra: getRaiz( dicionario: D));
    buscaPalavra( dicionario: D, palavra);
    strcpy(palavra, "Cadeira");
    removePalavra( dicionario: D, palavra);
    printf("\nPercorrimento em Ordem Alfabética\n");
    percorreEmOrdem( palavra: getRaiz( dicionario: D));
    return 0;
}
```

```
Percorrimento em pré-ordem
Caderno
Cadeira
Bola
Caso
Casa
Casamento
Casorio

Bola : Objeto usado em jogos

Percorrimento em Ordem Alfabética
Bola
Caderno
Casa
Casamento
Caso
Casorio
```

Main e saída do programa para quando a palavra **NÃO** é encontrada no dicionário.

```
int main() {
    char nomeArquivo[20] = "entrada.txt";
    char palavra[11] = "Casarao";
    dic *D = criaDic();
    carregaDicionario( dicionario: D, nomeArquivo);
    printf("\nPercorrimento em pré-ordem\n");
    percorrePreOrdem( palavra: getRaiz( dicionario: D));
    buscaPalavra( dicionario: D, palavra);
    strcpy(palavra, "Cadeira");
    removePalavra( dicionario: D, palavra);
    printf("\nPercorrimento em Ordem Alfabética\n");
    percorreEmOrdem( palavra: getRaiz( dicionario: D));
    return 0;
}
```

```
Percorrimento em pré-ordem
Caderno
Cadeira
Bola
Caso
Casa
Casamento
Casorio
Palavra não encontrada - Sugestões:
Caso
Casa
Casamento
Casorio

Percorrimento em Ordem Alfabética
Bola
Caderno
Casa
Casamento
Caso
Casorio
```

Dicas

Estudem as funções da biblioteca string.h

- strcmp
- strncpy
- strcpy

Para o uso do fgets, atentem-se aos caracteres de **retorno de carro** (carriage return) ao final da string, o que ocorre comumente ao ler arquivos de texto criados no Windows. Nesse sistema, as quebras de linha geralmente são marcadas com \r\n (retorno de carro e nova linha)