Aplicação Sobre Multilista Encadeada

Trabalho I da Disciplina Estrutura de Dados I

Lucas Oliveira Macedo e Nicolas Martins Gurgel

Professor: Gilmário Barbosa dos Santos

Novembro 2024

1 Enunciado

A partir de um arquivo de texto *arquivo.txt* com no mínimo 10 linhas de texto variando o conteúdo (não serve uma mera repetição da mesma palavra ou algo similar), sem formatação especial (negrito, itálico, underscore, etc) construa um programa que lê tal arquivo e carrega uma lista de listas dinâmicas duplamente encadeadas (*multiLDE*, e.g. Figura 1).

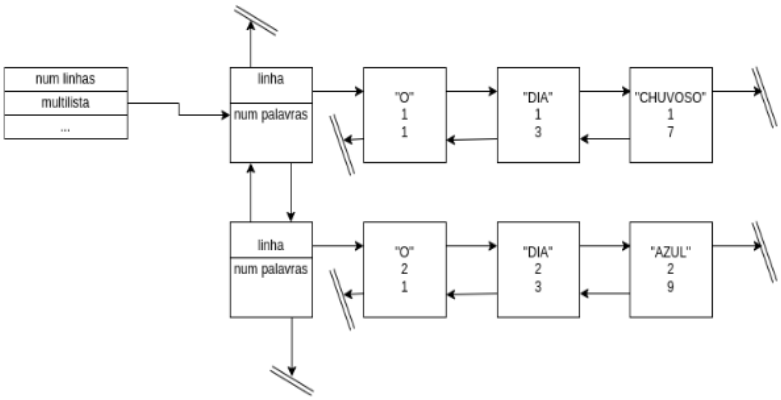

Figure 1: Estrutura da Multilista Duplamente Encadeada

2 Bibliotecas Utilizadas

Para este trabalho foram utilizadas bibliotecas padrões e uma biblioteca criada para facilitar a leitura, organização e manutenção do código conforme o projeto crescesse.

2.1 structs.h

Esta biblioteca contém todas as declarações e implementações das funções utilizadas no código juntamente com as structs necessárias para o funcionamento.

2.2 stdio.h

Nesta biblioteca estão localizadas as funções referentes às operações nas quais os mecanismos operam em função da entrada e da saída padrão, bem como em arquivos também.

2.3 stdlib.h

Nesta biblioteca estão localizadas as funções responsáveis pela manipulação da alocação de memória, e da desalocação também, bem como funções para converter números que estão representados em strings para algum tipo de dado responsável por representar números.

2.4 string.h

Nesta biblioteca estão contidos protótipos utilizados para a manipulação de strings. Estas funções são capazes desde a contagem, cópia e concatenação, comparação e diversas outras funções úteis para o atual projeto.

3 Funções Principais

3.1 Inserção de uma Nova Linha

Esta função inicializa uma linha em uma lista duplamente encadeada. É utilizada na primeira instanciação da lista baseada no arquivo fornecido.

```
noLinha* insereLinha(descLDE *lista, int numPalavras);
```

3.2 Inserção de Palavra

Esta função inicializa uma nova palavra em uma linha de uma lista duplamente encadeada. É utilizada na primeira instanciação de cada palavra de uma linha da lista duplamente encadeada.

```
int inserePalavra(noLinha *linha, char *palavra, int coluna);
```

3.3 Abertura de Arquivo

Esta função carrega um arquivo da memória nomeado *arquivo.txt*, pegando o texto do arquivo e transformado-o em uma multilista duplamente encadeada separada em linhas e palavras.

```
int abreArquivo(descLDE *lista, char *nomeArquivo);
```

3.4 Atualização de Modificações Feitas no Arquivo via Caixa de Diálogo

Esta função atualiza o arquivo.txt após as alterações realizadas (salvamento do conteúdo da multilista em arquivo). Cada palavra lida na lista de listas é gravada em um arquivo *arqTemp.txt*. Ao final, usando a função rename, o *arquivo.txt* é renomeado como *arquivoOLD.txt*, e após isso, renomeia *arqTemp.txt* como *arquivo.txt*.

```
void atualizaArquivo(descLDE *lista, char *nomeArquivo);
```

3.5 Exibição de Lista

Esta função é bem simples, ela apenas percorre por toda a multilista, imprimindo na tela todas as palavras inseridas na lista.

```
void exibeLista(descLDE *lista);
```

3.6 Busca de Palavra

Dada uma chave de busca, esta função irá buscar todas as ocorrências desta chave na multilista, informando todas as linhas e colunas quea chave aparece.

```
int buscaPalavra(descLDE *lista, char *chave);
```

3.7 Contagem de Palavras

Esta função apenas percorre a multilista incrementando um contador a cada palavra da lista, retornando assim a quantidade de palavras da multilista.

```
void contaPalavras(descLDE *lista);
```

3.8 Remoção de Todas as Ocorrências de uma Palavra

Dada uma multilista duplamente encadeada e uma chave de busca, esta função irá percorrer por toda a multilista, removendo todas as instâncias da chave encontradas ao longo do caminho.

```
void removePalavra(descLDE *lista, char *chave);
```

3.9 Remoção de Palavra na Linha e Coluna Especificadas

Dada uma multilista duplamente encadeada, uma chave de busca e uma coordenada (linha e coluna), esta função irá percorrer a multilista até chegar na coordenada e verificar se a chave de busca bate com a palavra encontrada na posição. Caso seja, a palavra é então removida.

```
void removePalavraPosicao(descLDE *lista, char *chave, int linha, int
    coluna);
```

3.10 Número de Ocorrências de uma Determinada Palavra

Dada uma multilista duplamente encadeada e uma chave de busca, esta função irá percorrer a multilista e incrementar um contador todas as vezes que esta palavra for encontrada na multilista, retornando assim todas as ocorrências da chave informada.

```
int numeroOcorrencias(descLDE *lista, char *chave);
```

3.11 Inserção de Palavra via Multilista

Dada uma multilista duplamente encadeada, uma palavra que o usuário deseja inserir e uma coordenada (linha e coluna), esta função irá inserir a palavra informada na multilista na coordenada especificada.

```
void inserePalavraLista(descLDE *lista, char *palavra, int indiceLinha,
     int indiceColuna);
```

3.12 Exibição de Linhas Contendo Pelo Menos uma Ocorrência de Determinada "Substring" Especificada Pelo Usuário

Dada uma multilista duplamente encadeada e uma *substring*, isto é, uma string contida em outra string, esta função irá percorrer a multilista informando todas as linhas em que a substring informada ocorre.

```
void substring(descLDE *lista, char *substring);
```

4 Funções Auxiliares

4.1 Leitura de Strings

Esta função foi criada para resolver o problema de buffer que ocorria ao utilizar diferentes métodos de leitura de input de usuário (*scanf*() e *gets*()). Esta função garante que não há um *carry* da última leitura no buffer, evitando assim problemas de leitura, especialmente de strings.

```
1 void lerString(char *string, int tamanho);
```

4.2 Limpeza de Tela

Função básica para limpar a tela, utilizada na função main para elaboração de menus intuitivos e para evitar excesso de informações na tela do terminal.

```
1 void limpaTela();
```

4.3 Press Enter

Função básica que aguarda uma confirmação do usuário (pressionar o botão enter) para continuar a execução do código. Utilizado na função main para o usuário conseguir ver modificações feitas e o andamento do menu.

```
1 void pressEnter();
```

4.4 Limpeza de Buffer

Função básica de limpeza de buffer, criada exclusivamente para uso na função *lerString*().

```
1 void limpaBuffer();
```

5 Compilação

A compilação do código é feita a partir do *GNU C Compiler* (*GCC*). Para isso é preciso que o GCC esteja instalado em sua máquina.

5.1 Verificar se o GCC Está Instalado na Máquina

Utilize o comando:

```
gcc --version
```

A mensagem que o terminal devolver será o suficiente para saber se o GCC está instalado ou não. Caso não esteja instalado, utilize o gerenciador de pacotes da sua distribuição Linux e procure pelo pacote do GCC:

5.1.1 Para Distribuições Basedas em Debian

```
sudo apt install build-essential
```

5.1.2 Para Distribuições Basedas em Fedora

```
sudo dnf install make gcc gcc-c++
```

5.1.3 Para Distribuições Basedas em Arch

```
sudo pacman -S gcc
```

Uma vez instalado, estando na pasta do arquivo fonte, basta utilizar o seguinte comando no terminal:

```
gcc -o [nome-executavel] main.c && ./[nome-executavel]
```

Não esqueça de substituir *nome-executavel* pelo nome que preferir que o arquivo executável tenha. A biblioteca structs.h e o arquivo.txt **devem estar na mesma pasta do arquivo fonte**.