Multilista Duplamente Encadeada

Estrutura de Dados I

André Luiz Correia Filho João Victor Zachêo

Professor: Gilmário Barbosa

Conteúdo

1 Descrição

Trabalho desenvolvido no curso Bacharelado em Ciência da Computação da UDESC - CCT.

O sistema consiste na representação e manipulação de um arquivo de texto em memória através de uma *Multilista Duplamente Encadeada (multiLDE)*

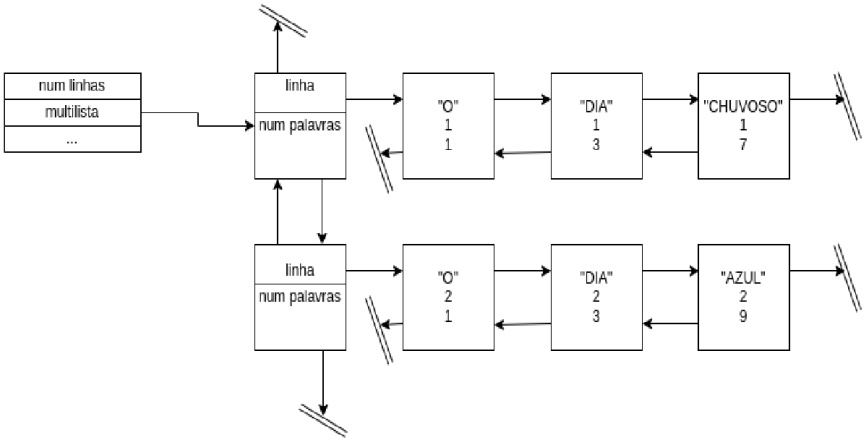

Figura 1: Representação da multilista duplamente encadeada utilizada no trabalho

2 Estrutura do Projeto

O projeto foi estruturado separando as funções em bibliotecas para melhor organização. Assim, cada biblioteca ficou responsável por funções específicas utilizadas no decorrer do projeto.

2.1 libEstrutura.h

Contém as funções responsáveis por manipular a multilista duplamente encadeada.

2.2 libMain.h

Contém as funções utilizadas na interface, leitura e escrita dos arquivos texto.

Além disso, foram utilizadas 4 *structs* para a construção da multilista que manipula os arquivos. Sendo elas:

Coordenada

```
typedef struct {
    int linha;
    int coluna;
}Coordenada;
```

Palavra

```
typedef struct informacoes{
    char palavra[tam_palavra];
    Coordenada coord;
    struct informacoes *frente;
    struct informacoes *tras;
}palavra;
```

Linha

```
typedef struct noMDE{
    palavra *palavras;
    struct noMDE *cima;
    struct noMDE *baixo;
    int numPalavras;
}linha;
```

Descritor

```
typedef struct multilista_duplamente_encadeada{
    int numLinhas;
    linha *multilista;
    char *caminhoArquivo;
}descritor;
```

3 Funcionalidades

3.1 Lê Arquivo

A função carrega um arquivo da memória que o usuário seleciona a partir de uma caixa de diálogo. Ela pega o texto do arquivo e transforma em uma multiLDE separada em linhas e palavras. Além disso, a função vincula a estrutura ao ponteiro do descritor passado como parâmetro.

```
void leArquivo(descritor *p);
```

3.2 Criar Backup

Quando chamada, a função cria um arquivo de backup antes das novas alterações serem salvas. O arquivo de backup é nomeado adicionando "OLD"ao final do nome do arquivo original.

```
1 void criaBackup(descritor *p);
```

3.3 Salva Alterações

Salva as alterações feitas no arquivo passando as informações da multiLDE para um arquivo de texto.

```
1 void salvaAlteracoes(descritor *p);
```

3.4 Exibir Texto

Apresenta ao usuário o texto estruturado na multiLDE.

```
1 void exibirTexto(descritor *p);
```

3.5 Busca Palavra

Procura todas as aparições de uma palavra no texto. A função retorna um vetor de coordenadas em forma de ponteiro contendo as coordenadas de todas as aparições da palavra inserida.

```
1 Coordenada* buscaPalavra(descritor *p,char s[],int *numCoords
    );
```

Parâmetros

- p - Descritor da multiLDE
- s[] - Palavra que o usuário deseja procurar
- numCoords - Quantidade de aparições encontradas

3.6 Remove Palavra

Remove todas as aparições de uma palavra no texto. Retorna 1 em caso de sucesso e 0 em caso de falha.

```
1 int removePalavra(descritor *p,char s[]);
```

3.7 Remove Coordenada

Remove a palavra localizada nas coordenadas passadas como parâmetro. Retorna 1 em caso de sucesso e 0 em caso de falha.

```
1 int removeCoordenada(descritor *p,int l,int coluna);
```

3.8 Exibir Total de Ocorrências de Palavra

Apresenta a quantidade de aparições de uma palavra no texto.

```
1 void exibirTotalOcorrenciasDePalavra(descritor *p,char s[]);
```

3.9 Exibir Total de Palavras

Apresenta a quantidade de palavras contidas no texto.

```
1 void exibirTotalPalavras(descritor *p);
```

3.10 Editar Palavra

Edita a palavra na coordenada passada como parâmetro.

```
1 void editarPalavra(descritor *p,int l,int coluna, char s[]);
```

3.11 Insere

Insere uma palavra na posição passada como parâmetro.

```
1 int insere(descritor *p,int l,int coluna, char s[]);
```

3.12 Procura Substring

Dada uma substring, a função apresenta as linhas que tenham palavras que contém a substring.

```
1 void procuraSubstring(descritor *p, char substr[]);
```

3.13 Printa Estrutura

Apresenta o texto indicando a quantidade de linhas no texto, a quantidade de palavras em cada linha e as coordenadas das palavras.

```
1 void printaEstrutura(descritor *p);
```

3.14 Reinicia Descritor

Reinicia o descritor, excluindo todas as linhas vinculadas ao ponteiro da *struct*.

```
1 void reiniciaDesc(descritor *p);
```

3.15 Reinicia Linha

Reinicia uma linha, excluindo todas as palavras vinculadas ao ponteiro da *struct*.

```
1 void reiniciaLinha(linha *l);
```

4 Compilação

O seguinte comando funciona apenas para a *main*, não está incluindo as *libs*. Sendo assim, serve apenas como exemplo de como compilar com as *flags* do *GTK*.

```
1 gcc arquivo.c `pkg-config --cflags --libs gtk+-3.0` && ./a.
    out
```

Para rodar a aplicação completa, usar sempre o **makefile**!

4.1 Tutorial

4.1.1 Compilação normal

```
1 make
```

4.1.2 Limpar arquivos compilados

```
1 make clean
```

4.1.3 Recompilar tudo

```
make rebuild
```

4.1.4 Ver informações de debug

```
make debug
```

4.1.5 Após o make

```
./programa
```

5 Bibliotecas Usadas

Além daquelas criadas para o projeto, algumas bibliotecas foram utilizadas a fim de facilitar a implementação do código. São elas:

- <stdio.h>
- <stdlib.h>
- <string.h>
- <gtk/gtk.h>

6 Autores

- João Zachêo
- André Correia