



SERVIÇO PÚBLICO FEDERAL · MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DE VIÇOSA · UFV
CAMPUS FLORESTAL

Trabalho 1 - AEDS 1

Dicionario

Arthur Teodoro [EF04672]

Saulo Victor [EF03336]

Henrique Campos [EF04673]

Florestal -

MG 2022

Sumário

1. Introdução	3
2. Organização	3
3. Desenvolvimento	4
3.1 Parte importante 1	4
3.2 Parte importante 2	4
4. Conclusão	5
5. Referências	5

1. Introdução

Foi proposto que fizéssemos um dicionário contendo as palavras que um texto possui e que iniciam com cada uma das letras do alfabeto, com o objetivo de colocar em prática a implementação de listas lineares e TADs. Para tratar o problema, utilizamos listas encadeadas com cabeça como estrutura de dados, além de TADs, seguindo as especificações do trabalho.

2. Organização

Na Figura 1 é mostramos como fizemos a organização do projeto. Na pasta **TP1** está a execução do projeto, separada em alguns módulos. A pasta **sources/** contém os arquivos .c dos TADs com as implementações dos mesmos e a pasta **headers/** contém os arquivos .h com as declarações dos TADs, junto com a estrutura de dados, lista encadeada, que foi solicitada.

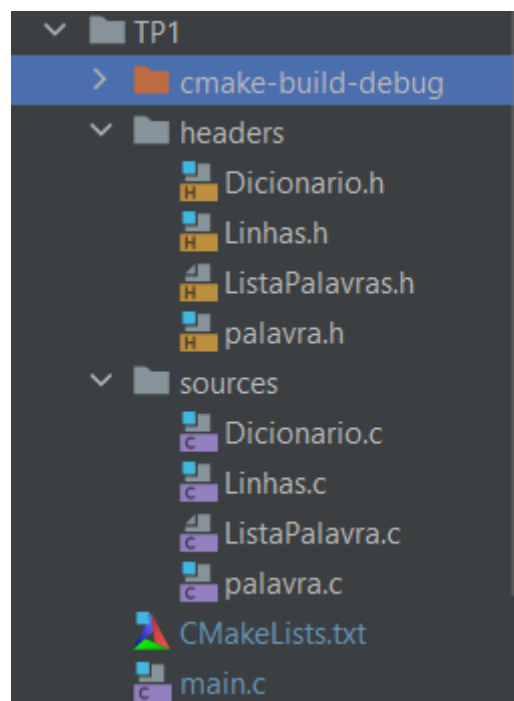


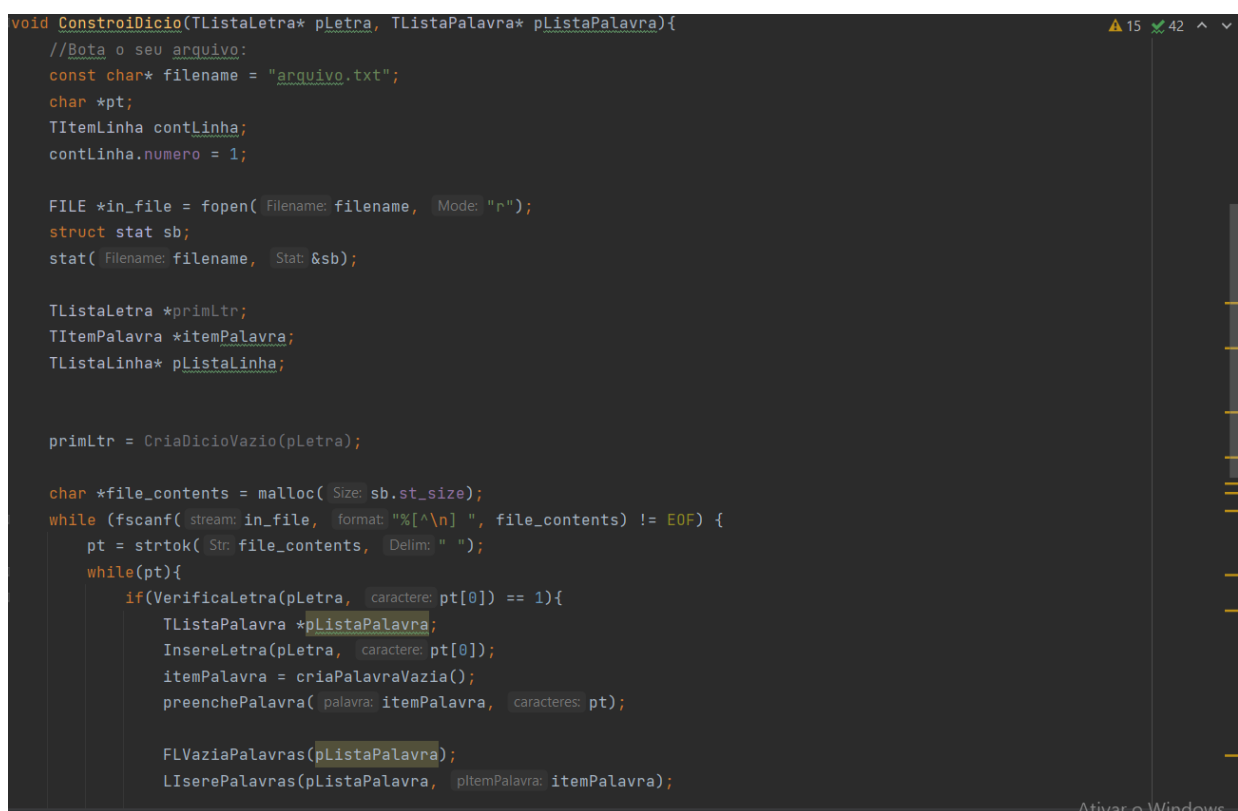
Figura 1 - Repositório do projeto.

Para executar o projeto, foi utilizado um arquivo **Makefile** com os comandos necessários para compilar e executar os códigos.

3. Desenvolvimento

Algumas partes importantes do projeto:

3.1 Parte importante 1



```
void ConstroiDicio(TListaLetra* pLetra, TListaPalavra* pListaPalavra){
    //Bota o seu arquivo:
    const char* filename = "arquivo.txt";
    char *pt;
    TItemLinha contLinha;
    contLinha.numero = 1;

    FILE *in_file = fopen( Filename: filename, Mode: "r");
    struct stat sb;
    stat( Filename: filename, Stat: &sb);

    TListaLetra *primLtr;
    TItemPalavra *itemPalavra;
    TListaLinha* pListaLinha;

    primLtr = CriaDicioVazio(pLetra);

    char *file_contents = malloc( Size: sb.st_size);
    while (fscanf( stream: in_file, format: "%[^\n] ", file_contents) != EOF) {
        pt = strtok( Str: file_contents, Delim: " ");
        while(pt){
            if(VerificaLetra(pLetra, caractere: pt[0]) == 1){
                TListaPalavra *pListaPalavra;
                InsereLetra(pLetra, caractere: pt[0]);
                itemPalavra = criaPalavraVazia();
                preenchePalavra( palavra: itemPalavra, caracteres: pt);

                FLVaziaPalavras(pListaPalavra);
                LiserePalavras(pListaPalavra, pitemPalavra: itemPalavra);
            }
            pt = strtok(NULL, " ");
        }
    }
}
```

Figura 2 - Função Constroi Dicionario

```

    FLVaziaLinhas(pListaLinha);
    LIserelinhas(pListaLinha, pItemLinha: contLinha.numero);
} else{
    if (VerificaPalavra(pListaPalavra, dadaPalavra: pt) == 1){
        itemPalavra = criaPalavraVazia();
        preenchePalavra( palavra: itemPalavra, caracteres: pt);
        FLVaziaPalavras(pListaPalavra);
        LIserPalavras(pListaPalavra, pItemPalavra: itemPalavra);

        FLVaziaLinhas(pListaLinha);
        LIserelinhas(pListaLinha, pItemLinha: contLinha.numero);
    }else{
        if (VerificaLinha(pListaLinha, atualLinha: contLinha) == 1){
            FLVaziaLinhas(pListaLinha);
            LIserelinhas(pListaLinha, pItemLinha: contLinha.numero);
        }else{
            break;
        }
    }
}

pt = strtok( Str: NULL, Delim: " ");
}

contLinha.numero = contLinha.numero+1;
}

```

Figura 3 - Parte 2 Função Constroi Dicionario

Essa parte do código foi importante, pois lê o arquivo, separa as palavras individualmente, utilizando a função strtok(), da biblioteca <string.h>. Além disso, chama as funções para preencher ou criar as TADs.

3.2 Parte importante 2

```

void imprimeDicio(TListaLetra* pLetra){
    const char* filename = "saida.txt";
    FILE *in_file = fopen( filename, Mode: "w");
    if(in_file == NULL){
        printf( format: "erro na leitura");
        exit( Code: 1);
    }
    TListaLetra* pAuxL;
    TListaLetra* pAuxL2;
    TListaLetra* pAuxL3;
    pAuxL = pLetra->primLtr->proxLetra;
    while(pAuxL != NULL){
        fprintf( stream: in_file, format: "=====");
        fprintf( stream: in_file, format: "Letra[%c]", pAuxL->primLtr->letter.caracter);

        pAuxL2 = pAuxL->primLtr->proxLetra->letter.ListaDaLetra.primP->proxPalavra;
        while (pAuxL2 != NULL){
            pAuxL3 = pAuxL->primLtr->proxLetra->letter.ListaDaLetra.primP->proxPalavra->Item.primL->proxLinha;
            fprintf( stream: in_file, format: "palavra: %s", pAuxL->primLtr->proxLetra->letter.ListaDaLetra.primP->proxPalavra->Item.string);

            while (pAuxL3 != NULL){
                fprintf( stream: in_file, format: "palavra: %d", pAuxL->primLtr->proxLetra->letter.ListaDaLetra.primP->proxPalavra->Item.primL->proxLinha->line.numero);
                pAuxL3 = pAuxL->primLtr->proxLetra->letter.ListaDaLetra.primP->proxPalavra->Item.primL->proxLinha->proxLinha;
            }
            pAuxL2 = pAuxL->primLtr->proxLetra->letter.ListaDaLetra.primP->proxPalavra->proxPalavra;
        }
        pAuxL = pLetra->primLtr->proxLetra->proxLetra;
    }
    fclose( File: in_file);
}

```

Figura 4 - Função imprime dicionário

Nessa parte do código, utilizamos ponteiros para acessar e imprimir os valores guardados dentro das TADs.

4. Conclusão

Concluimos que a utilização de lista encadeada, embora mais complicada, se torna mais eficiente que outras técnicas, por exemplo, lista linear com vetor, porque não precisamos ter um tamanho pré definido.

5. Referências

[1] Github. Disponível em: <<https://github.com/ArthurTBorges/TP1-AEDS-> >

[2] <http://linguagemc.com.br/arquivos-em-c-categoria-usando-arquivos/>

[3]

<https://www.arquivodecodigos.com.br/dicas/3189-c-como-usar-a-funcao-strtok-para-quebrar-uma-string-c-usando-delimitadores.html>