

## Universidade Federal de Viçosa – Campus Florestal

**Disciplina:** CCF 211 - Algoritmos e Estrutura de Dados 1

**Professora:** Thais Regina de Moura Braga Silva

**Alunos:** Marcos Biscotto - 4236, Alan Araújo - 5096, Gabriel Marques - 5097

### 1. Introdução

Para esse Trabalho Prático, foi-nos solicitado gerenciar a ocorrência de palavras em um texto. A ideia é montar um TAD Dicionário, armazenando as palavras que o texto possui em listas, ordenando e informando essas listas em ordem alfabética. Para isso utilizamos Listas Encadeadas com Cabeça, por meio de TAD's que representam os elementos do texto e listas que armazenam estruturas e outras listas, que por meio de manipulação de ponteiros e endereço de memória, conseguimos acessar e manipular seu conteúdo através de um menu interativo.

### 2. Organização

Em relação à organização e arquitetura do repositório do projeto, pensamos em deixá-los da forma mais simples possível, considerando que criar muitas pastas e deixar os arquivos do código dentro delas, poderia trazer complicações na hora da execução do programa. Um exemplo desse caso é que caso o arquivo de texto esteja localizado em uma pasta dentro da pasta do projeto, sua leitura seria feita através de comandos mais complicados do que no cenário em que o arquivo está explícito na pasta geral do arquivo.

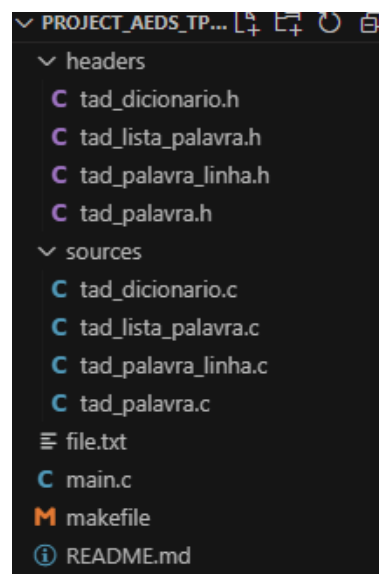
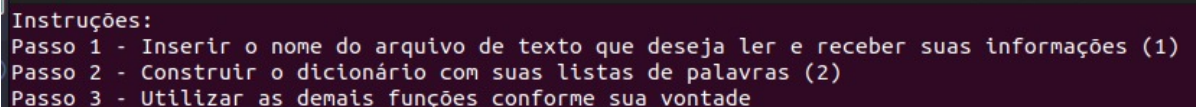


Figura 1 - Repositório do Projeto

Com base podemos ver na **Figura 1**, o arquivo de texto, makefile e o arquivo main do código estão todos juntos na pasta do projeto, e apenas os arquivos “.c” e “.h” dos TAD’s estão dentro de pastas. O arquivo de texto “file.txt” se refere ao texto que será passado como parâmetro para leitura, podendo esse receber qualquer nome.

Não acreditamos que seja necessário inserir instruções sobre programa e seu código dentro do arquivo “README”, uma vez que as variáveis, TAD’S e funções do código estarem nomeadas de forma autoexplicativa, além do mesmo estar devidamente comentado. Ainda assim, introduzimos uma opção no menu que serve para explicar como o programa funciona (**Figura 2**). O arquivo makefile possui os comandos de compilação e execução do programa, estruturado de maneira a compilar e rodar ao introduzir o comando “make” no terminal ou IDE.



```
Instruções:
Passo 1 - Inserir o nome do arquivo de texto que deseja ler e receber suas informações (1)
Passo 2 - Construir o dicionário com suas listas de palavras (2)
Passo 3 - Utilizar as demais funções conforme sua vontade
```

**Figura 2 - Instruções para o uso do programa.**

### 3. Desenvolvimento

Abaixo encontram-se descrições do processo de desenvolvimento do trabalho prático, desde sua interpretação até o que foi solicitado como produto final, além de pequenas descrições sobre as tomadas de decisões a respeito do desenvolvimento do programa.

#### 3.1) Interpretação:

Para esse trabalho prático, imaginamos um cenário onde uma grande estrutura que possui diversas listas encadeadas menores, que por sua vez armazenam estruturas que possuem cadeias de caracteres e uma outra lista encadeada que precisava constantemente ser acessada. Com isso em mente, partimos para pesquisas e a estruturação e segmentação do código.

#### 3.2) TAD’s:

Para atender às questões a nós demandadas, utilizamos da criação de mais do que apenas três TAD’s, pois dessa forma, mostrou-se mais fácil acessar cada “camada” do código, por meio de um efeito cascata. A utilização de apenas três TAD’s dificultava as operações envolvendo os outros componentes.

### 3.3) Tomada de Decisões:

Conforme progredimos, notamos que o principal desafio do programa era realizar a inserção da linha em que determinada palavra aparecia na lista, então decidimos gastar todo nosso esforço nessa parte, o que acabou nos custando uma grande quantidade de tempo e prejudicando o resultado final do TP.

### 3.4) O Dicionário:

A respeito da construção e exibição do dicionário, foi solicitado que fossem criadas listas apenas das letras do alfabeto as quais possuíam palavras no texto que começassem com tal letra. Porém, a maneira a qual pensamos que tal situação seria arquitetada era demasiada complicada e além de nossa capacidade, então escolhemos criar uma lista para cada letra do alfabeto e deixá-las vazias caso não aparecessem palavras com essas letras como suas iniciais.

## 4. Resultados

Como resultado, temos um menu interativo funcional (**Imagem 3**) que disponibiliza diversas opções para o usuário, recebe um comando do mesmo e realiza uma operação baseada na escolha do comando. A criação do dicionário funciona a partir da abertura, percorrimento e leitura de um arquivo de texto cujo nome será informado pelo usuário, recebendo e armazenando cada palavra em sua respectiva lista e contando as linhas do texto sempre que uma linha acaba. Recomenda-se deixar o arquivo de texto na mesma pasta que o arquivo “main.c” para uma leitura de forma a inserir apenas o nome do arquivo.

```
----- DICIONÁRIO DE AEDS -----
Opções do dicionário:
Nome do arquivo:
1. Inserir nome do arquivo
2. Construir o dicionário do texto
3. Exibir lista de palavras de determinada letra
4. Exibir toda a lista de palavras do texto
5. Verificar número de palavras do texto
6. Verificar se determinada palavra está na lista de palavras do texto
7. Remover determinada palavra de uma lista de palavras
8. Instruções para usar o programa
9. Sair do programa
-----
Opção desejada:
```

Imagem 3 - Menu Interativo

Caso o usuário escolha imprimir uma lista, essa lista pode variar de acordo com a opção solicitada pelo usuário. Uma dessas escolhas é a lista de palavras que começam com determinada letra do alfabeto, e para tal escolha a seguinte lista é impressa.

```
Insira uma letra: A
Letra |A| :
Palavra: a
Linhas: |1||2||3||4||5||9|
Palavra: apagou
Linhas: |3|
Palavra: aniversario
Linhas: |5|
Palavra: ate
Linhas: |7|
Palavra: ao
Linhas: |12|
Palavra: alem
Linhas: |13|
Palavra: animais
Linhas: |15|
Palavra: acabado
Linhas: |17|
```

Imagem 4 - Lista de palavras com a letra A

Vale ressaltar que a acentuação e caracteres da língua portuguesa estão aparecendo no sistema Linux. Caso o programa seja executado em um computador com o sistema operacional Windows sem nenhuma modificação, o resultado que aparecerá no terminal será algo como o exemplo abaixo.

```
----- DICIONÁRIO DE AEDS -----
Opções do dicionário:
Nome do arquivo: 
1. Inserir nome do arquivo
2. Construir o dicionário do texto
3. Exibir lista de palavras de determinada letra
4. Exibir toda a lista de palavras do texto
5. Verificar número de palavras do texto
6. Verificar se determinada palavra está na lista de palavras do texto
7. Remover determinada palavra de uma lista de palavras
8. Instruções para usar o programa
9. Sair do programa
-----
Opção desejada:
```

Imagem 5 - Execução em Windows

## 5. Conclusão

Por fim, com o término do trabalho, obtivemos um resultado que ao nosso ver atende às especificações anteriormente prescritas. Com a implementação de listas encadeadas, vetores, ponteiros, apontadores e funções da linguagem C como strcmp e strcpy, conseguimos realizar a leitura de um arquivo e inserir suas palavras em suas respectivas listas. O uso de listas encadeadas com cabeça foi o método utilizado pois acreditamos que essa forma se mostra mais simples e de mais fácil entendimento do que uma lista encadeada sem cabeça. O único ponto em que consideramos ter pecado, foi que não conseguimos deixar de fora a lista de letras que não possuem palavras, durante a impressão do arquivo dicionário.

## 6. Referências

- [1] Github. Disponível em: <<https://github.com/>> Último acesso em: 13 de outubro de 2022
- [2] Gitlab. Disponível em: <<https://gitlab.com/>> Último acesso em 11 de outubro de 2022
- [3] Stack Overflow. Disponível em <<https://stackoverflow.com/>> Último acesso em 14 de outubro de 2022
- [4] Geeks for Geeks. Disponível em <<https://www.geeksforgeeks.org/>> Último acesso em 9 de outubro de 2022
- [5] ZIVIANI, Nivio. **Projeto de Algoritmos com Implementações em Pascal e C**. 3ªEd. Cengage Learning, 23 junho 2010.