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

Trabalho 3 - AEDS 1

Dicionário Ordenado

Matheus Nascimento Peixoto [EF04662]
Matheus Nogueira Moreira [EF04668]
Pedro Augusto Martins [EF04692]

Florestal - MG
2022

Sumário

1. Introdução

Neste terceiro trabalho prático da disciplina Algoritmo e Estrutura de Dados (CCF211) é feito um aperfeiçoamento daquilo aplicado no trabalho prático 1 da mesma disciplina, apenas com uma pequena alteração na forma de funcionamento de um dos algoritmos, o “ListaPalavras”, e a implementação das formas de ordenação internas estudadas em sala de aula.

Este trabalho consiste no recebimento de um arquivo de texto, identificação das palavras nele presentes e suas respectivas linhas. Posteriormente é fornecida a opção de ordenação das listas de palavras criadas anteriormente através dos seguintes métodos de: *Bubble Sort*, *Selection Sort*, *Insertion Sort*, *Shellsort*, *Quicksort* e *Heapsort*. Tal ordenação é realizada em uma cópia da lista original, o que permite a visualização de ambas as listas mesmo após a ordenação, a criada logo após a abertura do arquivo como também a já ordenada. Após o processo de ordenação é mostrado o tempo gasto, em segundos, e o número de movimentações e comparações para cada letra da lista.

2. Organização

Para a melhor organização deste trabalho, seu código foi dividido entre as pastas Headers e Sources, com os cabeçalhos das funções e o desenvolvimento destas, respectivamente, além do arquivo “*main.c*”, “*makefile*” e dos textos utilizados para os testes.

A seguir é possível visualizar tal organização através de um “*print*” do VisualStudio Code:

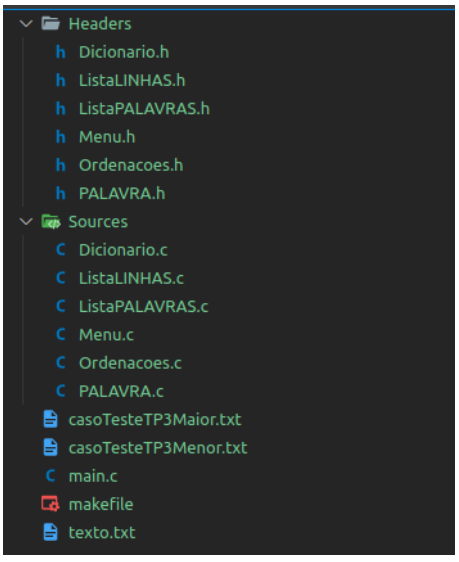

Figura 1: Organização do Trabalho Prático 03

3. Desenvolvimento

Para o desenvolvimento deste trabalho foram criados os seguintes arquivos:

3.1 - ListaLINHAS.h e ListaLINHAS.c

Utilizando de uma lista do tipo “arranjo” foi desenvolvido a lista das linhas das palavras presentes no texto.

3.2 - PALAVRA.h e PALAVRA.c

Tipo Abstrato de Dados (TAD) desenvolvido para receber e processar cada palavra presente no texto.

3.3 - ListaPALAVRAS.h e ListaPALAVRAS.c

Novamente utilizada uma lista do tipo “arranjo” dessa vez como foi exigido na especificação do trabalho, a qual, após verificar se determinada palavra não se encontra na lista, faz sua inserção juntamente das linhas ou apenas a inserção das linhas caso a palavra já esteja presente na lista.

3.4 - Ordenacoes.h e Ordenacoes.c

Nestes arquivos estão as funções de ordenações vistas em sala de aula durante a disciplina de CCF 211, sendo estas o Bubble Sort, Selection Sort, Insertion Sort, Shellsort, Quicksort e Heapsort.

Para cada algoritmo de ordenação existe contadores de movimentações e comparações os quais são incrementados à medida que acontecem trocas e comparações respectivamente.

Para possibilitar que após a ordenação seja possível visualizar as palavras de maneira desordenada novamente dentro de cada algoritmo é criada uma cópia da lista original para uma outra, a qual foi passada como parâmetro.

3.5 - Dicionario.h e Dicionario.c

O dicionário, criado a partir de uma lista simplesmente encadeada tem como função receber o arquivo de texto que possui as palavras que irão compor a lista, fazendo a identificação das palavras com suas respectivas linhas e organizar as listas com essas palavras de maneira alfabética, organização esta realizada na função “Insere_Nova_Lista_DICIONARIO”.

É nesse arquivo que estão as funções de impressão das listas de palavras que formam o dicionário da maneira que o usuário escolher, seja desordenadamente, isso é, a medida em que aparecem no texto original, ou ordenadamente, de acordo com um dos algoritmos disponíveis para escolha.

3.6 - Menu.h e Menu.c

Estes arquivos mostram ao usuário as opções disponíveis para escolha e direciona o que o usuário decidiu para as funções do Dicionário.

3.7 - Main.c

A fim de organização, o arquivo “*Main.c*” apenas chama a função “*Menu*” para executar todo o código.

4. Execução

Para a execução deste projeto existe o arquivo “*makefile*” para facilitar.
Basta digitar no terminal:

make all [Pressione Enter]

Posteriormente, caso esteja utilizando um computador Linux:

make Linux [Pressione Enter]

Ou, caso esteja em uma máquina com Windows, insira:

make Windows [Pressione Enter]

Após isso, com o programa já em execução, selecione primeiramente a opção 1, correspondente a opção para a criação do “Dicionário”. Logo em sequência escreva o nome do arquivo que contém o texto que será lido. Com o código estão três opções de texto “casoTesteTP3Maior.txt”, “casoTesteTP3Menor.txt”, os quais ambos foram disponibilizados para teste pelo Moodle aos alunos participantes desta disciplina, e um outro arquivo, este criado pelo grupo, que é bem menor que os demais e foi utilizado para melhor visualização durante alguns testes iniciais, denominado como “texto.txt”.

Depois de inserir o nome do arquivo de texto desejado, basta escolher alguma das demais opções apresentadas no “Menu”.

5. Resultados

Como resultado aparecem, quando é solicitada a impressão desordenada, as palavras, na ordem em que aparecem no texto inserido, juntamente com suas respectivas linhas. Quando solicitada a ordenação pelo usuário, o resultado obtido é o dicionário completo ou a lista completa com a letra desejada mostrando o tempo gasto na execução dessa ordenação bem como o número de comparações e movimentações para cada lista do dicionário.

Contudo algo chamou a atenção do grupo durante os testes. Para fins de melhor visualização e agilidade foi criado um outro arquivo, aos moldes do repassado pelo *Moodle*, o qual mostrava tudo conforme o esperado (as palavras com suas respectivas linhas). Mas ao testarmos o arquivo de texto que fizemos o download (o arquivo foi baixado e descompactado diretamente na pasta onde estava o código do trabalho) este mostrou que todas as palavras estavam na linha 1.

Em uma tentativa de descobrir o que estava causando essa diferença decidimos copiar o arquivo original, palavra por palavra, da mesma forma que estava, em um outro arquivo e testar. Para a surpresa do grupo este teste mostrou o resultado que esperávamos encontrar quando criamos o dicionário utilizando o arquivo original baixado do *Moodle*.

Para mostrar essa diferença gravamos um pequeno vídeo (4 minutos e 23 segundos) mostrando essa comparação.

Link do vídeo mostrando tal questão: https://youtu.be/Jm2RdFNfhuc

7. Referências

Para a realização deste trabalho utilizados alguns materiais que serão listados abaixo:

1 - Livro base da disciplina:

Projeto de Algoritmos com implementações em PASCAL e C - Nivio Ziviani