Universidade Federal de
Santa Catarina

Relatório do Trabalho Semestral

Algoritmos e Estruturas de Dados

Professor: Maiquel de Brito

Beatriz Fernanda, João Vitor, Lucas Tavares,
Viviane Block.

Blumenau, 23 de Novembro de 2020

1 - Introdução

A ideia principal tem como base a criação de uma lista que seria implementada a partir do conteúdo do arquivo de texto utilizado no programa. Onde depois de aberto, utilizaríamos uma função para que cada palavra presente nesse arquivo de texto viria a se tornar um novo elemento de uma lista encadeada, que armazenaria não somente a própria palavra em questão, como também contaria com a existência de um contador para registrar o número de ocorrências desta mesma palavra durante o decorrer do arquivo.

No final dessa etapa, já contando com uma lista existente e ordenada oriunda da função de ordenação, utilizaríamos funções específicas para as diferentes funcionalidades do programa, seja buscar palavras específicas ou listar todas elas de acordo com suas ocorrências. Essas funções seriam chamadas apenas caso o usuário optasse por determinada opção no menu.

2 - Implementação do código

2.1: A primeira etapa para a criação da base do código fonte consistiu na necessidade da existência dos parâmetros para formação de uma lista, ou seja, foram criadas as structs *"elemento"* e *"lstElemento",* juntamente com a declaração de uma lista inicialmente vazia, para que pudessem ser alocadas as palavras e contadores.

2.2: A segunda parte e talvez a mais importante do programa, foi o desenvolvimento da função que faria todo mecanismo de separação, contagem e ordenação das palavras do texto. Optamos por fazer todas estas etapas dentro de uma única função, para que as palavras pudessem ser organizadas imediatamente após serem alocadas na lista, e assim só passar para a próxima palavra, quando a anterior já estivesse em seu local definido.

2.3: Tendo então os elementos fundamentais para a criação da lista que seria implementada, o próximo passo se resumiu a criação da função *main* onde também foram feitas as declarações das variáveis necessárias para o início do programa, tais como a que armazena a escolha do usuário de acordo com as opções existentes no menu disponível logo abaixo das declarações. Este que executa a função solicitada pelo usuário por meio de um menu principal baseado em switch, implementado da seguinte forma:

- Opção 1: Pesquisa do número de ocorrências de uma determinada palavra escolhida pelo usuário, dentro do arquivo de texto;
- Opção 2: Faz a listagem de todas as palavras presentes no arquivo de texto de acordo com a quantidade de vezes que a mesma aparece;
- Opção 3: Encerramento do programa e apresentação dos créditos.

Foi feita a utilização de um "do while" englobando o "switch" para que o menu pudesse aparecer novamente após o término de cada operação realizada. Desta maneira, a opção três fica de fora do switch, interrompendo o laço e encerrando o programa quando assim desejado.

2.4: A busca é usada caso o usuário desejar fazer a consulta de uma palavra já inclusa na lista, apresentando a palavra buscada e o seu número de repetições.

2.5: Para satisfazer a segunda opção no menu principal, e imprimir a lista completa de palavras de acordo com seu número de aparições ao longo do texto, desenvolvemos a função “listagem”. Esta é responsável pela impressão da lista completa de palavras acrescidas de seus respectivos contadores.

3 - Funções

- Listagem(); : Chamada dentro do “case 2” para listagem completa de todas palavras contidas no texto e seus respectivos contadores com número de vezes em que cada uma delas aparece. Funciona utilizando um ponteiro “p” auxiliar equivalente a lista original de palavras e imprimi-as enquanto houver um próximo elemento na lista para satisfazer o “while”.

- Busca(); : Serve para funcionamento “case 1”, para buscar o número de ocorrências de uma palavra específica, determinada pelo usuário, no texto. A função recebe a palavra digitada e a compara, uma com as outras (utilizando “strcmp”), para encontrar a palavra e retorna-la junto de seu contador ao usuário.

- Le_ordena(); : É responsável por todo funcionamento principal do programa. Nela, é feita a separação, contagem, e ordenação simultânea de cada palavra do arquivo de texto inserido. A cada palavra, são feitas verificações para saber como adicionar e avançar o contador da palavra que está sendo inserida no momento, dependendo se é a primeira ocorrência ou uma palavra igual já foi previamente inserida.

- Insere_antes(); : Criada para auxiliar na inserção dos elementos em ordem dentro da lista, inserindo sempre elementos imediatamente antes daquele que atendeu as condições esperadas determinadas dentro da função “le_ordena”.

- LstElemento: Exibe o que compõe cada elemento da lista. Que no nosso caso seria o item (composto por: palavra e contador) e o ponteiro que vai apontar pro próximo elemento (nó).

4 - Complexidade

A complexidade atendida pelo programa atingiu os níveis máximos esperados para a leitura do arquivo e listagem das palavras (O(n)), entretanto, durante o desenvolvimento da função busca, apesar de termos a ciência de que o método que satisfaz a ordem exigida pelos critérios estabelecidos é o de busca binária, enfrentamos dificuldades ao implementar-lo devido a utilização de “strings”. Fazendo assim, com que optássemos pela busca linear, posteriormente adaptada para recursiva, com complexidade O(n).