**UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL**

**IMPLEMENTAÇÃO DO AUTOCOMPLETAR EM LINGUAGEM C COM ESTRUTURAS DE DADOS TIPO LISTA E ÁRVORE**

NATANAEL DA SILVA DEBONA - 00219823

RAFAEL JÚNIOR RIBEIRO- 00265830

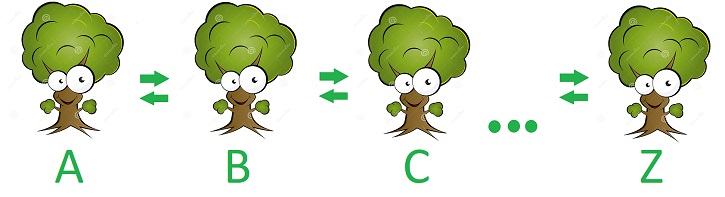
Porto Alegre, 27 de julho de 2017

**Introdução**

Este relatório tem como finalidade expor a escolha das estruturas utilizadas, bem como as funções implementadas, e apresentar os motivos que justificam tais escolhas.

A estrutura utilizada está dividida em 2 partes, Confecção do Dicionário, a qual recebe as palavras que serão usadas como base de dados, e a Confecção das previsões, a qual recebe os caracteres digitados e apresenta as possíveis palavras de previsão ao usuário.

A organização dos dados por parte da confecção do dicionário se deu através de uma lista duplamente encadeada circular, em que cada nodo representa uma árvore AVL.



**TAD do programa**

***struct lista***

*{*

*charletra\_index;*

*pNodoA \*letra\_arvore;*

*lista \*prox;*

*lista \*ant;*

*};*

***structTNodoA***

*{*

*char palavra[10];*

*int peso;*

*int FB; /\*Campo não usado na ABP\*/*

*pNodoA \*esq;*

*pNodoA \*dir;*

*};*

/\*Inicia a estrutura que irá receber as palavras do wikitionary. Uma floresta - lista - de árvores8\*/

***lista\* inicia\_floresta(lista \*inicio);***

/\*Na floresta haverá uma árvore para cada letra do alfabeto. Essa função inicia cada uma delas\*/

***lista\* inicia\_Arvore(lista \*floresta, char letra);***

/\*Função com o objetivo de pegar o endereço da árvore 'z', a árvore final da floresta\*/

***lista\* pega\_fim(lista \*inicio);***

/\*Calcula a altura. Útil para calcular o fator do nodo de uma árvore\*/

***int Altura (pNodoA \*a);***

/\*A partir da altura retornada, calcula o fator do nodo de uma árvore\*/

***intCalcula\_FB(pNodoA \*a);***

/\*Funções relacionada àoperação com arvores AVL\*/

***pNodoA\* rotacao\_direita(pNodoA \*pt);***

***pNodoA\* rotacao\_esquerda(pNodoA \*pt);***

***pNodoA\* rotacao\_dupla\_direita (pNodoA\* pt);***

***pNodoA\* rotacao\_dupla\_esquerda (pNodoA\* pt);***

***pNodoA\* Caso1 (pNodoA\* a, int \*ok);***

***pNodoA\* Caso2 (pNodoA \*a, int \*ok);***

***pNodoA\* insere\_na\_arvore(pNodoA \*a, char \*palavra,int peso, int \*ok);***

/\*Preparação para o início da inserção de dados na floresta. Chama a função insere\_na\_árvore

setando as variáveis necessárias para execução desta função\*/

***voidinsere\_na\_floresta(lista \*iniciofloresta,lista\*fimfloresta,char \*palavra,int peso);***

/\*Printa a árvore de uma letra. ´til para debugar o código\*/

***voidprintaArvore(pNodoA \*arvore\_letra);***

/\*Printa a floresta/dicionário. ´til para debugar o código\*/

***voidprintaFloresta(lista \*inicio);***

/\*Dado uma letra, retorna a árvore desta letra\*/

***lista\* busca\_arvore(lista \*iniciofloresta, lista \*fimfloresta,char \*palavra);***

/\*Dado um prefixo do arquivo de consulta, retorna uma árvore com TODAS as previsões desse prefixo.\*/

***pNodoA\* pega\_previsoes(lista \*iniciofloresta, lista \*fimfloresta,char \*palavra,char*** \*palavraLimite);

/\*Constrói a arvore de previsões procurando a partir da esquerda da árvore\*/

***pNodoA\* busca\_previsoesESQUERDA(pNodoA \*arvore,char \*palavra, char \*palavraLimite, pNodoA \*arvorePrevisoes,int \*flag);***

/\*Constrói a árvore de previsões procurando a partir da direita da árvore\*/

***pNodoA\* busca\_previsoesDIREITA(pNodoA \*arvore,char \*palavra, char \*palavraLimite,pNodoA \*arvorePrevisoes,int \*flag);***

/\*Insere efetivamente uma palavra do dicionário na árvore de previsões\*/

***pNodoA\* insere\_previsoes(pNodoA \*arvore, int peso, char \*palavra);***

/\*Escreve as previsões no arquivo de saída\*/

***voidescreve\_previsoes(FILE \*arquivoSaida,pNodoA \*arvorePrevisoes, int \*cont,intlimitePrevisoes);***

**Análise**

**1 - Criação do dicionário**

Para efetuar a montagem na memória das palavras recebidas pelo arquivo fornecido como .txt, o layout utilizado se deu na junção de duas estruturas de dados, uma lista duplamente encadeada e uma árvore AVL. Avaliamos também a possibilidade da implementação de apenas uma estrutura, uma árvore AVL englobando todas as palavras em uma única estrutura, notamos que deste modo teríamos uma busca mais ineficiente.

**1.1 - Lista duplamente encadeada**

Ao ser iniciado o programa, é efetuada a criação de uma lista duplamente encadeada, ordenada em ordem alfabética e com cada nodo representando uma árvore AVL, que por sua vez representa uma letra do alfabeto.

Como ideia inicial, tínhamos proposto utilizar uma ABP, porém, já estávamos utilizando uma ABP na consulta das previsões e uma AVL como nodos desta lista, então avaliamos que seria mais elegante utilizar uma LDE, pois abrangeria de maneira mais completa o conteúdo visto em aula. Além disto, a diferença no tempo de processamento geral do programa ao usar uma ABP no lugar da LDE não é significativa, visto que a lista é percorrida por ambos os lados, tanto no sentido a-> z quanto no sentido z->a.

**1.2 - Árvore AVL**

Com o intuito de realizar um número mínimo de comparações a cada inserção de uma nova palavra no dicionário, optamos pela estrutura de uma árvore. Ao estudar as possibilidades passadas, ABP, AVL, R-N e SPLAY baseamos a decisão final na que nos apresentasse a busca mais otimizada, que encontramos na AVL, pois esta apresenta melhor balanceamento dos nodos.

**2 - Formação das previsões**

Aqui fizemos uso de uma ABP, pois buscávamos a forma mais eficiente de inserir novas previsões, sem levar em consideração o balanceamento, pois este é desnecessário, tendo em vista que são visitados todos os nodos.

**Conclusão**

Ao realizarmos tal trabalho, conseguimos colocar em prática a teoria vista durante o semestre, passada pela professora Renata Galante, e solidificar na memória estes conhecimentos.

Vimos também que o recurso de autocompletar é bastante utilizado no dia-a-dia, em motores de busca na Web, serviços de mensagem de celular e editores de código e passamos a entender na prática o seu funcionamento.

Ao enfrentar os problemas que foram surgindo na codificação do programa, conseguimos aprimorar os nossos conhecimentos de depuração de código, além de entender, de forma simplificada, a confecção de um projeto corporativo.