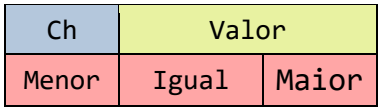
Algoritmos e Estruturas de Dados
Trabalho 4
Dicionário utilizando Ternary Search Tree (TST Trie)

Aluno: Marcos Oliveira de Sousa.

O presente trabalho foi todo desenvolvido na linguagem C e apresenta para o usuário um dicionário em que o mesmo pode adicionar palavras via um arquivo de texto e consultar prefixos.

Além disso, é possível consultar todas as palavras adicionadas, bem como remover determinadas palavras por meio de outro arquivo de texto.

O código foi desenvolvido utilizando como estrutura de dados a abordagem de árvore ternária da Trie, essa que é uma árvore de caracteres que pode ser utilizada para facilitar o acesso a palavras, seja por meio de buscas comuns (como buscar uma palavra completa), seja por meio de prefixos (buscando todas correspondências para demonstrar ao usuário se a palavra está contida na estrutura). A estrutura de uma TST Trie pode ser observada como:



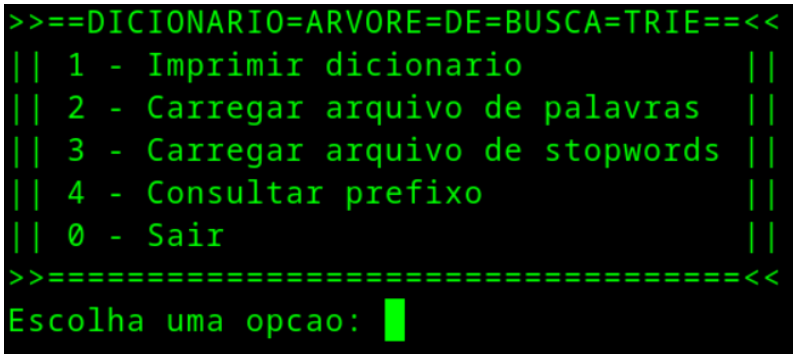
Onde "`Ch`" é o valor de caractere, "`Valor`" é um valor inteiro determinado atribuído a palavra final e os campos de "`Menor`", "`Igual`" e "`Maior`" são variáveis com o tipo de estrutura da árvore correspondentes aos valores ASCII do próximo caractere.

Por exemplo, a palavra "`cadeira`", se inserida no início, tem toda sua cadeia de caracteres inseridas no meio, já que essa fora inserida antes de qualquer outra palavra, já a palavra "`cadeia`", quando inserida, irá percorrer o caminho do meio até o momento de 'quebra' de sequência, que nesse caso seria a letra '`a`' em contraste com a letra '`r`' da palavra "`cadeira`".

Em relação ao valor atribuído, não ficou claro no documento sobre qual seria o critério para atribuir determinado valor a uma palavra completa, sendo o critério adotado a de nova palavra. A cada nova palavra, um valor iterador atribui um novo valor a nova palavra, começando do 0 nesse caso. E caso removida alguma palavra, o valor iterador não volta ao valor removido, o mesmo continua a iterar.

Importante ressaltar que, como pressuposto acima, os valores de caracteres aceitos no programa são correspondentes a tabela ASCII, sendo assim, não é recomendável inserir caracteres de fora do intervalo de letras da tabela – que nesse caso seriam 65 a 90 para letras maiúsculas e 97 a 122 para letras minúsculas. E como já pressuposto novamente, somente letras são aceitas, símbolos ou números acabam por desconsiderar a entrada no dicionário, importante que o usuário tome o devido cuidado com espaços extras que não sejam apenas quebras de linhas, palavras que contenham tabulação ou espaços serão desconsideradas, sendo aceitas somente palavras que estejam empilhadas linha a linha.

Segue abaixo o menu do usuário e uma explicação sobre cada uma das funcionalidades:



1. Imprimir dicionário: função que permite com que o usuário confira todas as palavras adicionadas ao dicionário ao lado de seu valor correspondente (que nesse caso seria ordem de inserção)
2. Carregar arquivo de palavras: função que permite com que o usuário possa adicionar novas palavras por meio de um arquivo de texto, claro, seguindo as recomendações já descritas nos textos acima.
3. Carregar arquivo de *stopwords*: função que permite com que o usuário delete palavras contidas no dicionário, importante ressaltar que, palavras que não estão contidas são somente ignoradas, enquanto as removidas são mostradas ao usuário no momento de sua exclusão.
4. Consultar prefixo: permite com que o usuário consulte determinada(s) palavra(s) por meio de um determinado prefixo, supondo que temos as palavras "`carro`", "`carrete`" e "`caminhonete`" inseridas no dicionário, caso o prefixo inserido seja "`ca`", serão impressas todas as palavras, caso o prefixo seja "`car`", apenas os dois primeiros serão impressos ao usuário.

Sobre a parte técnica do menu, foram adicionados pequenos detalhes ao usuário para pausar o terminal durante alguma impressão ou limpar a tela após alguma seleção de opção.

```
85  // Funcao para pausar por 5seg no linux
86  // Pre-condicao: nenhuma
87  // Pos-condicao: se windows, usa o pause
88  void pause5()
89  {
90      #ifdef _WIN32
91      system("pause");
92      #elif __linux__
93      system("sleep 5");
94      #endif
95  }
96
97  // Funcao para limpar terminal
98  // Pre-condicao: nenhuma
99  // Pos-condicao: nenhuma
100 void cls()
101 {
102     #ifdef _WIN32
103     system("cls");
104     #elif __linux__
105     system("clear");
106     #endif
107 }
```

Sendo esses detalhes puramente estéticos a fim de melhorar a experiência do usuário, possuindo opção para LINUX e Windows de controle de terminal.

Pra finalizar, sobre a parte técnica do código, é importante salientar apenas que, a fim de evitar processo de remoção que possa não resultar em uma remoção de palavras, fora implementado uma função de busca para verificar se uma palavra existe ou não no dicionário, para além disso, verificar o código disponível no anexo.