Relatório Prático de Estruturas de Dados e Algoritmos I

João Fernandes - 32409 , Leonardo Catarro - 43025

Junho 2021

Conteúdo

1 Introdução

Para a compomente prática da unidade curricular de Estruturas de Dados e Algoritmos I foi-nos pedido para criar uma aplicação que utilizando o método de escrita inteligente T9Keys crie mensagens curtas, ou seja, a cada número de 2 a 9 está associado entre 3 a 4 letras do alfabeto latino, neste caso o português e inglês, e também o uso de um dicionário que permita introduzir palavras com os números associados.

Esta aplicação irá permitir criar uma mensagem escrevendo uma palavra de cada vez, em que são recebidos os algarismos correspondentes às letras dessa palavra e oferecendo sugestões até ser encontrada a palavra desejada, caso contrário é pedido ao utilizador para introduzir a palavra que deseja e sendo depois essa palavra adicionada ao dicionário.

Para este trabalho não foi necessário a utilização de sinais de pontuação e de espaços na escrita das palavas, sendo usado as teclas de 2 a 9, a tecla 1 e a tecla 0 para a conclusão da mensagem e para encerrar o programa, respetivamente.

2 Tipos de dados abstratos usados

2.1 Tabela de Hash

Para este trabalho foram usadas duas HashTable com hashing aberto(cadeias separadas), usando listas ligadas de forma a nos ser possível ter palavras que sejam compostas pela mesma sequência de keys no mesmo índice da HashTable.

Um das tabelas foi criada para armazenar os caracteres das T9Keys e a segunda para guardar as palavras do dicionário usado.

Esta TAD consiste em fazer hashing aberto aplicando o hash às chaves e sendo as suas operações feitas numa cadeia externa, neste caso listas ligadas. Tendo várias vantagens como sendo fácil de implementar, a tabela nunca fica cheia ,sendo possível adicionar mais elementos na cadeia e é maioritarimente usada quando ou com que frequência as chaves são inseridas ou removidas da tabela.

2.2 Listas Ligadas

As listas foram usadas, como foi acima mencionado, apenas para servir de cadeia separada das HashTables.

3 Principais funções

3.1 FILE *OpenDictionary(char *fileName, char* wayToOpen)

Recebe como argumentos o nome do dicionário e a forma de abertura do mesmo(escrita, leitura, etc..). Função que abre o dicionário previamente criado, recorrendo à função fopen(), desde que esteja guardado na pasta dictionaries.

Retorna o ficheiro passado como argumento.

3.2 void CloseDictionary(FILE *fp)

Recebe como argumento um dicionário. Função que vai fechar o dicionário anteriormente aberto, recorrendo á função fclose().

3.3 void UpdateDictionary(char* source, char *destination, wchar_t *word)

Recebe como argumentos o nome do dicionário atualmente usado, o nome do dicionário de destino(dicionário atualizado) e a palavra que deve ser inserida no novo dicionário. Esta função começa por abrir ambos os ficheiros, com a ajuda da função OpenDictionary, de seguida copia todo o conteudo do ficheiro source para o ficheiro destination, caso este esteja vazio, e por ultimo insere a nova palavra no dicionário atualizado. Caso, este dicionário source seja um dicionário com frequência, é adicinada a palavra com frequência 0, através da função InsertWordAccordingFrequency.

3.4 wchar_t *CleanWordProcess(wchar_t* word)

Recebe como argumento a palavra à qual vai ser aplicada o processo de "limpeza". Função que vai iterar a palavra passada por argumento, convertendo-a para caratéres minuscúlos(até encontrar um hifen ou um apóstrofe), com a ajuda da função towlower() da biblioteca wctype, guardando esses caratéres num array auxiliar.

Retorna o array auxiliar, que contem a palavra limpa.

3.5 unsigned long StringToIntAccordingT9Keys(wchar_t *word, HashTable KeysTable)

Recebe como argumentos a palavra que vai ser convertida e a tabela de hash das T9Keys. Função que tal como indicado no seu nome irá transformar a string(palavra) introduzida em inteiros(números) de acordo com as T9Keys previamente definidas (2 a 9).

Retorna a sequencia de inteiros que codifica a palavra.

3.6 void ProcessData(FILE *fp, HashTable KeysTable, HashTable WordsTable)

Recebe como argumentos o dicionário, e as Tabelas de Hash. Função que carrega as T9Keys (2 a 9) para a HashTable das Keys, com os caractéres que cada uma representa e de seguida são inseridos no dicionário as palavras(iterando pelo dicionário, linha a linha), recorrendo às funções CleanWordProcess e StringToIntAccordingT9Keys.

3.7 wchar_t *InsertWordInHashAndPhrase(FILE *fp, wchar_t *wordToInsert, wchar_t *phrase, HashTable WordsTable, HashTable KeysTable)

Recebe como argumentos o dicionário, a palavra a ser inserida na phrase e na Tabela de Hash, a frase atual e ambas as HashTables, das T9 Keys e das Palavras. Esta função começa por dar append da word á phrase, com a ajuda da função wcscat(), semelhante a strcat(), mas para o tipo de dados wchar_t. De seguida, verifica o tipo do dicionário, executa o processo de "limpeza"da palavra e, por ultimo, insere-a na Tabela de Hash.

Retorna a palavra que foi inserida.

3.8 void InsertWordAccordingFrequency(wchar_t * Key, long wordFreq, long wordInInt, HashTable H)

Recebe como argumentos a palvra a inserir, a sua frequencia de utilização, a sua conversão para keys e a Hashtable onde a palavra vai ser inserida. Esta função é em tudo semelhante ao InsertWords(função que insere elementos na hashtable), apenas com a particularidade de recorrer à função SortLinkedList para ordenar a lista onde foi inserida Key.

3.9 void SortLinkedList(List L)

Recebe como argumento uma Lista Ligada. Função presente em InsertWordsAccordingFrequency que, para o dicionário de frequências, após a inserção da palavra vai ordenar a lista do indice onde esta foi inserida por ordem decrescente. Esta ordenação é similar ao BubbleSort.

4 Principais variáveis

4.1 FILE *fp

Variável que usa um apontador de ficheiro (fp) para manusear e manter registo do ficheiro que está a ser acessado no momento em que o programa é inicializado.

4.2 wchar_t phrase[BUFFER_LENGTH]

Array que tem como objetivo guardar as sugestões aceites pelo utilizador.

4.3 char *argv[1]

Argumento passado ao programa que contém o nome do ficheiro a ser utilizado.

4.4 clock begin / clock end

Variável que vai inicializar a contagem do tempo de execução do programa (clock begin) e a variável que vai parar de contar esse mesmo tempo (clock end).

4.5 HashTable KeysTable / HashTable WordsTable

Variáveis para as Tabelas de Hash usadas no desenvolvimento do trabalho.

5 Problemas enfrentados no desenvolvimento do trabalho

5.1 Uso de dados do tipo wchar_t

Com o uso de acentos e hífen na língua portuguesa geriu-se a dificuldade em criar e/ou manusear certas palavras , como por exemplo "diz-se", que em UTF-8 na linguagem C não é possível representar e usando wchar_t que permite aumentar a gama de caractéres representáveis para UTF-16, permitindo assim, se fosse necessário, criar mais dicionáios para outras línguas, como japonês.

5.2 Atualizar o dicionário com palavras do tipo wchar_t

Na nossa tentativa inicial de atualizar o dicionário surgia o problema em que ao ser introduzida uma nova palavra para o dito dicionário o programa ou simplesmente não o fazia ou as apagava no momento em que as palavras previamente copiadas.

5.3 Alocação de memória

C sendo uma linguagem muito rigorosa com a alocação de memória, ao ser usada a função malloc(), alocação dinâmica de memória, criou casos em que devido à nossa má, ou até mesmo à falta de alocação de memória, o programa não agia como seria de esperar, criando até em certos momentos bugs que não eram suposto estarem lá.

5.4 Falha de leitura da última linha do ficheiro

Ao serem implementadas as funcionalidades extra deparámo-nos com um erro que consistia na falha de leitura da última linha dos dicionários.

6 Conclusão

Inicialmente quando foi lançado o enunciado deste trabalho, a criação desta aplicação não parecia ser algo complicado de se fazer, mas conforme foram sendo implementadas as funcionalidades surgiam novos problemas e na resolução desses mesmos problemas ocasionalmente surgiam outros.

Para concluir, apesar dos problemas encontrados pelo caminho do desenvolvimento da aplicação foi-nos possível implementar todas as funcionalidades desejadas pela professora, incluindo as funcionalidades adicionais, e submeter a aplicação dentro do prazo estipulado.