UNIVERSIDADE DE ÉVORA

Estruturas de Dados e Algoritmos I
2020-2021

Trabalho Prático
Serviço de Mensagens Curtas Inteligente

Trabalho realizado por:

- Tomás Dias nº42784
- Diogo Pinto nº43044

Estrutura da aplicação

Na fase inicial do desenvolvimento da aplicação, pensou-se em como guardar a informação dos dicionários de forma eficiente e que permitisse um acesso rápido ao conteúdo dos mesmos.

Por isso, optou-se por utilizar a estrutura de dados **Hashtable** com hashing **aberto** para gestão de colisões, utilizando **Linked Lists** para o efeito.

As funções que implementam e manipulam as **Linked Lists**, contidas no ficheiro **list.c**, são as seguintes:

- **List CreateList(List L)**: cria uma LinkedList apenas com um nó header.
- **List MakeEmptyList(List L)**: esvazia a LinkedList L.
- **bool IsEmpty(List L)**: verifica se a LinkedList L está vazia.
- **bool IsLast(Position P, List L)**: verifica se o nó na posição P da LinkedList L é o último.
- **Position FindList(ElementType X, List L)**: retorna a posição do elemento X na LinkedList L.
- **Position FindPrevious(ElementType X, List L)**: encontra a posição do elemento anterior a X na LinkedList L.
- **void InsertList(ElementType X, List L, Position P)**: insere o elemento X na posição P da LinkedList L.
- **void DeleteList(ElementType X, List L)**: apaga o elemento X da LinkedList L.
- **void RemoveList(List L)**: apaga a LinkedList L da memória.
- **Position Header(List L)**: retorna o Header da LinkedList L.
- **Position First(List L)**: retorna o primeiro nó da LinkedList L depois do Header.
- **Position Advance(Position P)**: retorna a posição do nó seguinte.
- **ElementType Retrieve(Position P)**: retorna o valor do nó que está na posição P.

- **void PrintList(List L)**: faz print da LinkedList L.

As funções que implementam e manipulam a Hashtable, contidas no ficheiro **hashsep.c,** são as seguintes:

- **static bool isPrime(int n)**: verifica se o inteiro n é primo ou não.
- **static int NextPrime(int n)**: retorna o número primo seguinte ao n.
- **Index Hash(long int digits, int TableSize)**: função que vai fazer o Hashing tendo em conta a chave que recebe e o tamanho da tabela.
- **HashTable InitializeTable(int TableSize)**: função que inicializa a HashTable com tamanho TableSize.
- **void DestroyTable(HashTable H)**: vai apagar todas as LinkedLists ligadas à Hashtable e depois apaga a Hashtable.
- **void RetrieveList(long int code, HashTable H)**: retorna uma LinkedList da HashTable.
- **Position Find(ElementType Key, HashTable H)**: encontra o elemento Key na HashTable H e retorna a sua posição.
- **void Insert(ElementType Key, HashTable H)**: insere uma palavra nova à HashTable H.
- **HashTable Delete(ElementType X, HashTable T)**: apaga o elemento X da HashTable (acede à LinkedList correspondente e apaga o elemento).
- **HashTable MakeEmpty(HashTable T)**: percorre a HashTable e esvazia todas as suas LinkedLists.
- **void PrintHashTable(HashTable T)**: faz print da HashTable (print a todas as LinkedLists).

Para a implementação da dita aplicação, implementou-se as seguintes funções auxiliares contidas no ficheiro **t9lib.c**:

- **int CountWords(char const* const filename)**: retorna o número de palavras presentes no dicionário.

- **void BuildKeyboard(ElementType Keyboard[])**: constrói o teclado com os algarismos e as respetivas letras associadas.

- **long int Encode(wchar_t word[], ElementType Keyboard[])**: retorna o código da palavra recebida.

- **void LoadDictionary(char const* const filename, Hashtable H, ElementType Keyboard[])**: carrega o dicionário e guarda-o na hashtable.

Funcionamento da aplicação

O programa começa por ler um ficheiro **txt** de input que contém todas as palavras do dicionário de forma a calcular o número de palavras existentes.

Depois, é inicializada a **Hashtable** onde todas as palavras do dicionário vão ser armazenadas. O tamanho da **Hashtable** vai ser igual ao número primo seguinte ao número de palavras existentes no dicionário calculado anteriormente.

De seguida, é carregado o dicionário para a **Hashtable**. Cada elemento armazenado na **Hashtable** é composto pela palavra a ser guardada e o seu código em algarismos. Este código é depois utilizado para determinar o índice da **Hashtable** em que o elemento poderá ser acedido. O elemento é adicionado à cabeça da **Linked List** associada ao índice. Este processo é repetido para todas as palavras do dicionário sendo medido o tempo de execução do mesmo.

Após o carregamento do dicionário, é pedido ao utilizador para inserir um código.

Se o código for composto apenas pelo algarismo **1**, a mensagem final é mostrada no ecrã.

Se o código for composto apenas pelo algarismo **0**, o programa termina.

Caso não se verifiquem as duas situações anteriores, o código introduzido é utilizado para localizar o índice da **Hashtable** e a respetiva **Linked List** a ser acessada.

Após isso, serão sugeridas as palavras com o mesmo código introduzido presentes na **Linked List** ao utilizador e este pode dizer ‘s’ se é a palavra que quer adicionar à sua mensagem final e ‘n’ caso contrário.

Se o programa já tiver sugerido todas as palavras da **Linked List** e o utilizador não tiver aceitado nenhuma é pedido ao utilizador para escrever ele próprio a palavra no teclado. Depois de escrever a palavra, esta é adicionada à **LinkedList** para usos futuros no mesmo programa e é adicionada à mensagem final.

É pedido novamente um código ao utilizador, e assim sucessivamente, até que este saia do programa.

Informações adicionais

O trabalho é composto pelas seguintes diretorias:

- ***dictionaries***: onde estão contidos os dicionários disponibilizados.
- ***include***: onde estão contidos os ficheiros header (.h) onde se encontram as definições de funções, estruturas de dados e data types utilizados na implementação da aplicação.
- ***src:*** onde se encontra o código fonte da aplicação e as respetivas funções implementadas.

Para **compilar** deve utilizar o seguinte comando:

~$ gcc src/list.c src/hashsep.c src/t9lib.c src/mcurtas.c -o mcurtas

Para **executar** deve utilizar o seguinte comando:

~$ ./mcurtas dictionaries/[dicionário que pretende carregar].txt

De referir que as funcionalidades adicionais **não** foram implementadas.