



**INSTITUTO POLITÉNICO NACIONAL**

**ESCUELA SUPERIOR DE CÓMPUTO**

**ESTRUCTURAS DE DATOS**

**PRÁCTICA #3**

**DICCIONARIO CON HASHING ABIERTO**

**GRUPO: 1CM8**

**EQUIPO: LAS MÁS PERRAS**

**INTEGRANTES:**

* JIMÉNEZ DELGADO LUIS DIEGO 2019630461
* SÁNCHEZ CASTRO AARÓN GAMALIEL 2019630079
* SÁNCHEZ TIRADO CITLALI YASMÍN 2019630096

**PROFESOR:** EDGARDO ADRIÁN FRANCO MARTÍNEZ

**FECHA DE ENTREGA:** 22 DE ABRIL 2019

**INTRODUCCIÓN**

**MARCO TEÓRICO**

**PLANTEAMIENTO DEL PROBLEMA**

Con la implementación del TAD lista realizar la implementación de una tabla hash abierta, capaz de soportar el almacenamiento de palabras y sus definiciones (Diccionario de palabras). La función hash a usar deberá ser analizada por cada equipo y deberá de justificarse (determinar al menos dos y reportarlas).

**DISEÑO Y FUNCIONAMIENTO DE LA SOLUCIÓN**

Se propone la creación de un TAD Tabla Hash (*TADHashTable*) utilizando el TAD Lista Doblemente Ligada, de modo que un usuario de la librería *TADHashTable* no tiene conocimiento explícito que esta librería utiliza la lista doblemente ligada.

AGREGAR DIBUJO

El *TADHashTable* cuenta con las siguientes operaciones.

* InicializarTabla:
* DestruirTabla:
* AgregarATabla:
* EliminarDeTabla:
* elemento BuscarEnTabla:
* ModificarTabla:
* Estadisticas:
* AplicarHash

**IMPLEMENTACIÓN**

/\*

FUNCIÓN: caracterLatino(int c)

RECIBE: int c (VALOR ASCII DE UN CARACTER)

DEVUELVE: int

DESCRIPCIÓN: ESTA FUNCIÓN RECIBE EL VALOR ASCII DE UN CARACTER QUE MUESTRA UN SÍMBOLO EN LA CONSOLA Y LO CONVIERTE A SU CORRESPONDIENTE CARACTER

OBSERVACIONES:

\*/

int caracterLatino**(**int c**){**

**switch(**c**){**

**case** 177**:** **return** 164**;** //ñ

**break;**

**case** 145**:** **return** 165**;** //Ñ

**break;**

**case** 193**:** **return** 181**;** //Á

**break;**

**case** 201**:** **return** 144**;** //É

**break;**

**case** 205**:** **return** 214**;** //Í

**break;**

**case** 211**:** **return** 224**;** //Ó

**break;**

**case** 218**:** **return** 233**;** //Ú

**break;**

**case** 225**:** **return** 160**;** //á

**break;**

**case** 233**:** **return** 130**;** //é

**break;**

**case** 237**:** **return** 161**;** //í

**break;**

**case** 243**:** **return** 162**;** //ó

**break;**

**case** 250**:** **return** 163**;** //ú

**break;**

**case** 241**:** **return** 164**;** //ñ

**break;**

**case** 209**:** **return** 165**;** //Ñ

**break;**

**case** 252**:** **return** 129**;** //ü

**break;**

**default:** **return** c**;** //RETORNA EL MISMO CARACTER

**break;**

**}**

**}**

La función caracterLatino hace la corrección de los caracteres (indicados en los comentarios) que se encuentran en una cadena de texto o en un archivo de texto, a través de esta conversión es posible representar los caracteres de manera adecuada en pantalla.

/\*

FUNCIÓN: cargarDiccionario(tablaHash \*diccionario).

RECIBE: int (DIRECCIÓN DE MEMORIA DE LA TABLA HASH A OPERAR).

DEVUELVE: NADA.

DESCRIPCIÓN: SE ENCARGA DE EXTRAER LOS CARACTERES CONTENIDOS EN UN ARCHIVO DE TEXTO, SEPARANDO LA PALABRA DE SU DEFINICIÓN. POSTERIORMENTE

AGREGA ESTE PAR DE CADENAS A UN ELEMENTO Y ES AGREGADO A UNA LISTA DENTRO DE LA TABLA HASH INDICADA EN EL PARÁMETRO.

OBSERVACIONES:

\*/

void cargarDiccionario**(**tablaHash **\***diccionario**){**

FILE **\***archivo**;**

registro nuevaPalabra**;**

char rutaArchivo**[**100**],**palabra**[**100**],**palabraAlt**[**100**],**definicion**[**251**],**estado**=**'p'**;**

/\* ES NECESARIOO CREAR LA VARIABLE palabraAlt PUESTO QUE ALGUNAS PALABRAS TIENEN "ALTERNATIVAS"

MARCADAS POR EL SIMBOLO /, EL PROGRAMA DEBE RECONOCERLAS COMO PALABRAS DIFERENTES

LA VARIABLE estado INDICA LA TRANSICIÓN DE PALABRA A DEFINICIÓN

\*/

int charArchivo**,**i**=**0**;**

printf**(**"Ingresa la ruta del archivo:\n>"**);**

scanf**(**"%s"**,** **&**rutaArchivo**);**

archivo **=** fopen**(**rutaArchivo**,**"r"**);**

palabraAlt**[**0**]** **=** '#'**;**

**while((**charArchivo **=** getc**(**archivo**))** **!=** EOF**){**

charArchivo **=** caracterLatino**(**charArchivo**);**

**if(**charArchivo **!=** 195**){**

**if** **(**charArchivo**==**'/' **&&** estado **==** 'p'**){**//ESTA PALABRA TIENE UNA VERSIÓN ALTERNATIVA

estado**=**'q'**;**

palabra**[**i**]** **=** '\0'**;**

i**=**0**;**

**continue;**

**}**

**if** **(**charArchivo**==**':'**){**//COMIENZA LA DEFINICIÓN

**if(**estado **==** 'p'**)**//SE TRATA DE UNA PALABRA SIN ALTERNATIVAS

palabra**[**i**]** **=** '\0'**;**

**if(**estado **==** 'q'**)**//SE TRATA DE LA ALTERNATIVA DE ALGUNA PALABRA

palabraAlt**[**i**]** **=** '\0'**;**

estado**=**'s'**;**

i**=**0**;**

**continue;**

**}**

**if** **(**charArchivo**==**'\n'**){**//SI OCURRE UN SALTO DE LÍNEA ES EL FINAL DE LA DEFINICIÓN

definicion**[**i**]=**charArchivo**;**

definicion**[**i**+**1**]** **=** '\0'**;**//SE CIERRA LA DEFINICIÓN

strcpy**(&**nuevaPalabra**.**palabra**[**0**],&**palabra**[**0**]);**//LA PALABRA SE COPIA AL ATRIBUTO palabra DEL NUEVO registro

strcpy**(&**nuevaPalabra**.**definicion**[**0**],&**definicion**[**0**]);**//LA PALABRA SE COPIA AL ATRIBUTO palabra DEL NUEVO registro

AgregarATabla**(**diccionario**,**nuevaPalabra**);**

printf**(**"\nPalabra %s agregada exitosamente!"**,**nuevaPalabra**.**palabra**);**

**if(**palabraAlt**[**0**]** **!=** '#'**){**

strcpy**(&**nuevaPalabra**.**palabra**[**0**],&**palabra**[**0**]);**

strcpy**(&**nuevaPalabra**.**definicion**[**0**],&**definicion**[**0**]);**

AgregarATabla**(**diccionario**,**nuevaPalabra**);**

printf**(**"\nPalabra %s agregada exitosamente!"**,**nuevaPalabra**.**palabra**);**

palabraAlt**[**0**]** **=** '#'**;**

**}**

fflush**(**stdout**);**

estado **=** 'p'**;**

palabra**[**0**]** **=** '#'**;**

i**=**0**;**

**continue;**

**}**

**if(**estado **==** 'p'**){**//ESCRIBE LA PALABRA NORMAL

palabra**[**i**]=**charArchivo**;**

i**++;**

**}**

**if(**estado **==** 'q'**){**//COMIENzA A ESRIBIR LA PALABRA ALTERNA

palabraAlt**[**i**]=**charArchivo**;**

i**++;**

**}**

**if(**estado **==** 's'**){**//COMIENZA A ESCRIBIR LA DEFINICIÓN

definicion**[**i**]=**charArchivo**;**

i**++;**

**}**

**}**

**}**

printf**(**"\n\nPor favor, espere..."**);**

sleep**(**2**);**

system**(**"cls"**);**

//system("clear");

**return;**

**}**

La función cargar diccionario es capaz de leer un archivo de texto carácter por carácter, cada uno de estos es enviado a la función caracterLatino para una posible corrección y una vez formada una palabra con su respectiva definición son agregadas a un registro (elemento definido como registro), este registro es agregado a su correspondiente lista a través de la función AgregarATabla. Este proceso se repite hasta que el carácter leído del archivo sea igual al carácter *End Of File* (EOF).

/\*

FUNCIÓN: nuevaPalabra(tablaHash \*diccionario).

RECIBE: int (DIRECCIÓN DE MEMORIA DE LA TABLA HASH A OPERAR).

DEVUELVE: NADA.

DESCRIPCIÓN: SE ENCARGA DE SOLICITAR UNA NUEVA PALABRA CON SU RESPECTIVA DEFINICIÓN, AÑADE ESTAS CADENAS A UN NUEVO ELEMENTO Y ÉSTE ES AGREGADO

AL DICCIONARIO INDICADO EN EL PARÁMETRO.

OBSERVACIONES: NO SE VALIDA SI EL USUARIO INTRODUCE UNA PALABRA CON ALGUNA ALTERNTIVA. EJEMPLO: PIEDRA/ROCA

\*/

void nuevaPalabra**(**tablaHash **\***diccionario**){**

registro nueva**;**

char palabra**[**100**],**definicion**[**255**];**

int aux**;**

printf**(**"Ingresa la nueva palabra:\n>"**);**

scanf**(**"%s[a-zA-Z ]"**,&**palabra**);**

fflush**(**stdin**);**

//LIMPIANDO CARACTERES

**for(**aux**=**0**;**aux**<**strlen**(**palabra**);**aux**++){**

palabra**[**aux**]=**caracterLatino**(**palabra**[**aux**]);**

**}**

printf**(**"Ingresa su definicion:\n>"**);**

scanf**(**"%[a-zA-Z ,.]s"**,&**definicion**);**

fflush**(**stdin**);**

//LIMPIANDO CARACTERES

**for(**aux**=**0**;**aux**<**strlen**(**definicion**);**aux**++){**

definicion**[**aux**]=**caracterLatino**(**definicion**[**aux**]);**

**}**

strcpy**(&**nueva**.**palabra**[**0**],&**palabra**[**0**]);**

strcpy**(&**nueva**.**definicion**[**0**],&**definicion**[**0**]);**

AgregarATabla**(**diccionario**,**nueva**);**

printf**(**"Palabra %s agregada exitosamente!\n"**,**palabra**);**

printf**(**"Por favor, espere...\n"**);**

sleep**(**2**);**

system**(**"cls"**);**

//system("clear");

**return;**

**}**

La función nuevaPalabra solicita al usuario que introduzca una nueva palabra para ingresar en la tabla, posteriormente solicita su definición y estas dos cadenas son agregadas a un registro. Dicho registro es enviado como parámetro a la función AgregarATabla para su inserción en la tabla hash.

/\*

FUNCIÓN: buscaPalabra(tablaHash \*diccionario).

RECIBE: int (DIRECCIÓN DE MEMORIA DE LA TABLA HASH A OPERAR).

DEVUELVE: NADA.

DESCRIPCIÓN: SE ENCARGA DE SOLICITAR LA PALABRA QUE SE DESEA BUSCAR, ÉSTA ES INGRESADA A UN NUEVO ELEMENTO TEMPORAL Y ENVIADA A LA FUNCIÓN

CORRESPONDIENTE DE LA LIBRERÍA TADHashTable.h

OBSERVACIONES: EL USUARIO NO DEBE INTRODUCIR PALABRAS ALTERNATIVAS EN LA MISMA CADENA. NO ES NECESARIO SOLICITAR LA DEFINICIÓN.

\*/

void buscaPalabra**(**tablaHash **\***diccionario**){**

registro buscame**,**encontrada**;**

char palabra**[**100**],**null**[**4**]=**"NULL"**;**

int aux**,**index**;**

printf**(**"Ingresa la palabra a buscar:\n>"**);**

scanf**(**"%s[a-zA-Z ]"**,**palabra**);**

**for(**aux**=**0**;**aux**<**strlen**(**palabra**);**aux**++){**//LIMPIAMOS LA CADENA POR ESO DE LOS CArACTERES RAROS

palabra**[**aux**]=**caracterLatino**(**palabra**[**aux**]);**

**}**

strcpy**(&**buscame**.**palabra**[**0**],&**palabra**[**0**]);**//COPIAMOS LA PALABRA INGRESADA A LA PALABRA DEL NUEVO ELEMENTO

encontrada**=**BuscarEnTabla**(**diccionario**,**buscame**);**

**if(**strcmp**(&**encontrada**.**palabra**[**0**],&**null**[**0**])!=**0**){**

printf**(**"%s: %s\n"**,**encontrada**.**palabra**,**encontrada**.**definicion**);**

**}**

system**(**"pause"**);**

system**(**"cls"**);**

//system("clear");

**return;**

**}**

La función buscaPalabra solicita al usuario que introduzca la palabra de la cual desea consultar su definición, esta palabra es añadida a un registro auxiliar y enviado a la función BuscarEnTabla para que regrese la definición de dicha palabra en caso de encontrarse en el diccionario.

/\*

FUNCIÓN: eliminarPalabra(tablaHash \*diccionario).

RECIBE: int (DIRECCIÓN DE MEMORIA DE LA TABLA HASH A OPERAR).

DEVUELVE: NADA.

DESCRIPCIÓN: SE ENCARGA DE SOLICITAR LA PALABRA QUE SE DESEA ELIMINAR, LA ALMACENA EN UN NUEVO ELEMENTO TEMPORAL Y LA ENVÍA A LA RESPECTIVA

FUNCIÓN DE LA LIBRERÍA TADHashTable.h

OBSERVACIONES: EL USUARIO NO DEBE INTRODUCIR PALABRAS ALTERNATIVAS EN LA MISMA CADENA. NO ES NECESARIO SOLICITAR LA DEFINICIÓN.

\*/

void eliminarPalabra**(**tablaHash **\***diccionario**){**

registro eliminame**;**

char palabra**[**100**];**

int aux**,**index**;**

printf**(**"Ingresa la palabra a eliminar\n>"**);**

scanf**(**"%s[a-zA-Z ]"**,&**palabra**);**

**for(**aux**=**0**;**aux**<**strlen**(**palabra**);**aux**++){**//LIMPIAMOS LA CADENA POR ESO DE LOS CArACTERES RAROS

palabra**[**aux**]=**caracterLatino**(**palabra**[**aux**]);**

**}**

strcpy**(&**eliminame**.**palabra**[**0**],&**palabra**[**0**]);**//COPIAMOS LA PALABRA INGRESADA A LA PALABRA DEL NUEVO ELEMENTO

printf**(**"Eliminando...\n"**);**

EliminarDeTabla**(**diccionario**,**eliminame**);**

printf**(**"Eliminada exitosamente!\nPor favor espere...\n"**);**

sleep**(**2**);**

system**(**"cls"**);**

//system("clear");

**return;**

**}**

La función eliminarPalabra solicita la usuaria que introduzca la palabra que desea eliminar del diccionario, esta cadena es agregada a un registro auxiliar y enviada a la función

/\*

FUNCIÓN: modificarDefinicion(tablaHash \*diccionario).

RECIBE: int (DIRECCIÓN DE MEMORIA DE LA TABLA HASH A OPERAR).

DEVUELVE: NADA.

DESCRIPCIÓN: SE ENCARGA DE SOLICITAR UNA PALABRA Y SU NUEVA DEFINICIÓN, ESTAS CADENAS SON AGREGADAS A UN NUEVO ELEMENTO Y ENVIADAS A LA RESPECTIVA

FUNCIÓN DE LA LIBRERÍA TADHasTable.h

OBSERVACIONES: NO SE VALIDA SI EL USUARIO INTRODUCE UNA PALABRA CON ALGUNA ALTERNTIVA. EJEMPLO: PIEDRA/ROCA

\*/

void modificarDefinicion**(**tablaHash **\***diccionario**){**

registro eModificado**;**

char palabra**[**100**],**nuevaDefinicion**[**255**];**

int aux**,**index**;**

printf**(**"Ingresa la palabra a modificar su definicion\n>"**);**

scanf**(**"%s[a-zA-Z ]"**,&**palabra**);**

**for(**aux**=**0**;**aux**<**strlen**(**palabra**);**aux**++){**//LIMPIAMOS LA CADENA POR ESO DE LOS CArACTERES RAROS

palabra**[**aux**]=**caracterLatino**(**palabra**[**aux**]);**

**}**

strcpy**(&**eModificado**.**palabra**[**0**],&**palabra**[**0**]);**//COPIAMOS LA PALABRA INGRESADA A LA PALABRA DEL ELEMENTO ACTUAL

printf**(**"Ingresa la nueva definicion:\n>"**);**

scanf**(**"%[a-zA-Z ,.]s"**,&**nuevaDefinicion**);**

**for(**aux**=**0**;**aux**<**strlen**(**nuevaDefinicion**);**aux**++){**//LIMPIAMOS LA CADENA POR ESO DE LOS CArACTERES RAROS

nuevaDefinicion**[**aux**]=**caracterLatino**(**nuevaDefinicion**[**aux**]);**

**}**

strcpy**(&**eModificado**.**definicion**[**0**],&**nuevaDefinicion**[**0**]);**

printf**(**"Modificando...\n"**);**

ModificarTabla**(**diccionario**,**eModificado**);**

printf**(**"Modificada exitosamente!\nPor favor espere...\n"**);**

sleep**(**2**);**

system**(**"cls"**);**

//system("clear");

**return;**

**}**

**FUNCIONAMIENTO**

**POSIBLES MEJORAS**

**CONCLUSIONES**

**ANEXOS**