Grup 15
David Gayete NIA: 204921
Eduard Masip NIA: 207322

Practica 1 – EDA 2

GRUP15

Objetivo a resolver

1. Crear un diccionario que implemente una HashTable con un algoritmo Hash.

2. La HashTable contiene un array de 26 LinkedList, cada una está destinada a almacenar palabras que empiezan por la misma letra.
(26 letras del abecedario)

3. Una LinkedList, almacena Nodes, que son estructuras del tipo WordInfo encapsuladas con un puntero al siguiente WordInfo que se guarde en la misma LinkedList (empieza por la misma letra), y al anterior.

4. Que no repita palabras si ya están guardadas.

5. Que elimine palabras del diccionario dejando bien enlazados los Nodes consecutivos al eliminado.

6. Imprimir todas las palabras de una LinkedList

7. Imprimir todo el diccionario ordenado.

Solución

1- El algoritmo de Hash utiliza una string que contiene todas las letras del abecedario, y con un 'for' empieza a comparar la primera letra de la palabra recibida como parámetro con cada letra del abecedario. Cuando coinciden, devuelve el valor de la 'i', que es la posición de la letra en el abecedario. La función encargada de hacer este proceso es la **"hash_function(char* word)"**.

2- La HashTable se inicializa llamando a la función **"init_list(LinkedList* l)"** por cada una de las LinkedList que contiene. (Una por letra en el abecedario).

3- Cada una de las WordInfo se pasa a la función **"encapsulate(LinkedList* l, WordInfo wi)"** junto con la LinkedList a la que tiene que ir y esta función encapsula la estructura WordInfo dentro de la estructura Node. Ahí se enlazará junto al last nodo que hay en la misma parte del array de LinkedList.

4- Para hacer que no haya palabras repetidas, utilizaremos la función **"exists_word(HashTable* table, char* word)"**, donde buscará la palabra en la lista que le pertenece, y si es encontrada, no la añade, por lo contrario la añade.

Grup 15
David Gayete NIA: 204921
Eduard Masip NIA: 207322

5- Para eliminar palabras enlazando bien los nodos, primero miraremos si la palabra escogida pertenece al diccionario o no con la función **“exists_word(HashTable* table, char* word)”**. Si pertenece, llamará a la función **“delete_word(HashTable* table, char* word)”** que será la encargada de eliminar y dejarlo enlazado correctamente.

6- Cuando hay que imprimir todas las palabras de una LinkedList, se llama a la función **“print_list(LinkedList* l)”** ,donde se le pasa la lista que se quiere imprimir, y esta función la recorre todos los nodos a través de la variable first de la LinkedList y la variable ‘next’ de cada nodo, hasta encontrar que el next es nulo imprimiendo toda la información que contiene la WordInfo.

7- Para imprimir el diccionario ordenado, creamos un array de tamaño fijo, con esta función **“size(HashTable* table)”** que devuelve cuántas palabras hay en el diccionario, luego se copian todas las palabras en este array nuevo, recorriendo todas las LinkedList del diccionario **“vectoritzar(char** toSort, HashTable* dict)”**.
Una vez tenemos el array de Strings con todas las palabras, llamamos a la función **“sort(char** toSort, int size)”** que se encarga de ordenar el Array por en orden ascendente usando el algoritmo de inserción y imprime cada vez que hace un cambio en el orden del Array dejándolo finalmente totalmente ordenado.

Implementación

Las estructuras de datos utilizadas son la HashTable, el Node, la LinkedList y la WordList.

Estas estructuras ya venían dadas, pero les hemos añadido algunos campos para que fuera más fácil implementar el diccionario.

```
typedef struct {
    LinkedList list[N];
    int size;
} HashTable;

typedef struct {
    char word[MAX_WORD_LENGTH];
    char definition[MAX_DEFINITION_LENGTH];
    char pos;
} WordInfo;
```

Grup 15
David Gayete NIA: 204921
Eduard Masip NIA: 207322

```
typedef struct _Node {
    WordInfo data;
    struct _Node* next;
    struct _Node* prev;
} Node;

typedef struct {
    Node* start;
    Node* last;
    int size;
} LinkedList;
```

Ejemplos

AÑADIR

En el caso de añadir una palabra nueva, primero imprimiremos la lista a la que pertenecerá y luego imprimiremos la lista con la palabra nueva.
Para implementar la función de añadir, utilizaremos este código:

```
print_list(&dict.list[0]);
WordInfo inf;
strcpy(inf.definition,"tiene que ver con el espacio");
inf.pos='A';
strcpy(inf.word,"aerospacial");
insert_word_info(&dict , inf);
printf("nueva lista\n");
print_list(&dict.list[0]);
```

Imprimimos la lista:

```
-> algorithm (N): set of rules specifying how to solve some problem
-> analyze (V): consider in detail and subject to an analysis
```

Vemos que la lista de la letra ‘a’ contiene dos palabras. Ahora añadiremos la nueva, llamada aerospacial.

```
-> algorithm (N): set of rules specifying how to solve some problem
-> analyze (V): consider in detail and subject to an analysis
nueva lista
-> algorithm (N): set of rules specifying how to solve some problem
-> analyze (V): consider in detail and subject to an analysis
-> aerospacial (A): tiene que ver con el espacio
```

Como podemos ver, la palabra se ha añadido correctamente y la imprime con las demás.

Grup 15
David Gayete NIA: 204921
Eduard Masip NIA: 207322

ELIMINAR

En el caso de eliminar alguna palabra del diccionario, primero imprimiremos la lista a la que pertenece, eliminaremos la palabra y posteriormente, volveremos a imprimir la lista.
Utilizaremos este código:

```
print_list(&dict.list[2]);
delete_word(&dict,"computer");
printf("nueva lista\n");
print_list(&dict.list[2]);
```

Imprimimos la lista:

```
-> compute (V): make a mathematical calculation or computation
-> computer (N): a machine for performing calculations automatically
```

Ahora eliminamos la palabra computer y volvemos a imprimir la lista:

```
-> compute (V): make a mathematical calculation or computation
-> computer (N): a machine for performing calculations automatically
nueva lista
-> compute (V): make a mathematical calculation or computation
```

Como podemos ver, la palabra computer se ha eliminado de la lista y únicamente nos queda compute.

LIMPIAR DICCIONARIO

Para el caso de vaciar el diccionario, primero imprimiremos el diccionario ordenado alfabéticamente y posteriormente la volveremos a imprimir, donde podremos ver que estará vacío.
Utilizaremos este código:

```
print_sorted_word_info(&dict);
clear_table(&dict);
printf("nuevo diccionario");
print_sorted_word_info(&dict);
```

Imprimimos el diccionario:

```
algorithm
analyze
compute
computer
efficient
exponentially
fast
optimal
procedure
program
search
slow
study
```

Grup 15
David Gayete NIA: 204921
Eduard Masip NIA: 207322

Ahora imprimimos el nuevo diccionario vacío:

```
algorithm
analyze
compute
computer
efficient
exponentially
fast
optimal
procedure
program
search
slow
study

nuevo diccionario
Done!
```

Donde vemos que, efectivamente, está vacío.