



Instituto Politécnico Nacional
ESCUELA SUPERIOR DE CÓMPUTO



ESTRUCTURAS DE DATOS

PROFESOR: FRANCO MARTÍNEZ EDGARDO ADRIÁN

1CM9

PRÁCTICA 03. DICCIONARIO CON HASHING ABIERTO

Tania Itzel Núñez García

Verónica Jackeline Quiros Díaz

Erick Efraín Vargas Romero

Semestre: 2016 – 2017 “A”

20 de Noviembre de 2016

ÍNDICE

INTRODUCCIÓN	3
PLANTEAMIENTO DEL PROBLEMA	3
DISEÑO Y FUNCIONAMIENTO DE LA SOLUCIÓN	4
IMPLEMENTACIÓN DE LA SOLUCIÓN	5
FUNCIONAMIENTO	6
ERRORES DETECTADOS	12
POSIBLES MEJORAS	12
CONCLUSIONES	12
ANEXO	14
BIBLIOGRAFÍA	18

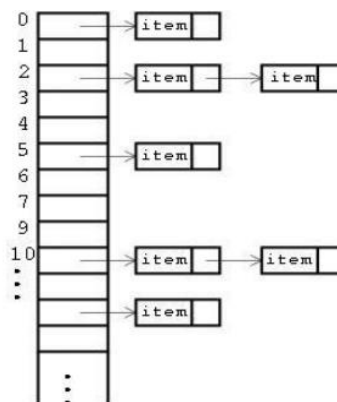
INTRODUCCIÓN

Contar con estructuras de datos es importante tanto para el buen manejo de la información, como para la buena ejecución de los programas de cómputo. Un problema dentro de algunos archivos es que es posible almacenar grandes cantidades de información que generalmente no está bien organizada o bien, es muy extensa, lo que dificulta el acceso a un dato en particular. Esto se incrementa de manera exponencial cuando pasamos del manejo de archivos a bases de datos, que pueden contener millones de veces la cantidad de información contenida en un archivo, con miles o millones de registros que en su momento serán utilizados.

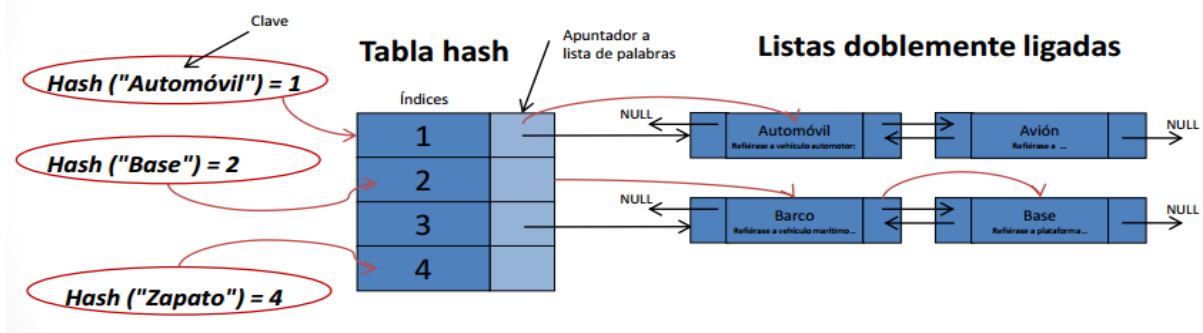
Existe el grupo de búsquedas por transformación de llaves (Hash), que aumenta la eficiencia, en cuanto al tiempo de ejecución, ya que accede a los registros por lo general más rápidamente, pero va a depender de su implementación. Dicho esto, se elaboró la práctica y presente reporte, para comprender y evaluar conceptos previamente vistos en clase y sacar conclusiones al respecto.

PLANTEAMIENTO DEL PROBLEMA

Con la implementación del TAD lista se debe 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

De acuerdo al problema planteado, se analizó y se llegó al acuerdo de que se elaborara la función hash para resolver más eficaz y eficientemente el problema. Para su elaboración, se consideró que es un algoritmo o transformación que, aplicado a la clave, devuelve la posición del destino en donde podemos ubicar o recuperar el elemento que contiene dicha clave. Y que normalmente consta de tres partes:

- Transformación: Si la clave no es numérica se convierte en un número. Con frecuencia se utiliza el valor ASCII de cada carácter y luego se aplican operaciones matemáticas para obtener un número entero.
- Generación: El número generado se procesa con un algoritmo que trata de uniformizar la distribución de las claves en el rango de direcciones.
- Compresión: Se comprime el número obtenido multiplicándolo por un factor para adecuarlo a la capacidad de almacenamiento disponible. La función Hash debe definirse al momento de diseñar el sistema y su selección tiene gran incidencia en rendimiento del sistema. Una buena función Hash debe tener las siguientes características:
 - Sencilla, de manera que sea fácil de codificar y minimice el tiempo de cálculo.
 - Distribución Uniforme de las direcciones tratando que la generación distribuya en forma aleatoria las claves y evite agrupamientos.
 - La idea es seleccionar una función que permita obtener una distribución con el mayor grado de uniformidad posible para evitar colisiones.

Después de tomar en cuenta estas características para crear la función, se realiza lo siguiente:

- Se mostrara un menú principal con las siguientes opciones:

Diccionario Hash

- 1.- Cargar un archivo de definiciones
- 2.- Agregar una palabra y su definición
- 3.- Modificar una definición
- 4.- Eliminar una palabra
- 5.- Salir

1 - Cargar un archivo de definiciones • Realiza la carga en la tabla hash de un archivo que contiene una palabra y su definición en cada línea del archivo.

2.- Agregar una palabra y su definición • Permite agregar una palabra nueva al diccionario y su definición.

3.- Modificar una definición • Permite que se busque una palabra y modifique su definición

4.- Eliminar una palabra • Permite buscar una palabra y eliminarla.

IMPLEMENTACIÓN DE LA SOLUCIÓN

En base a la solución planteada, para la implementación se creó la función de hashing con ayuda de la lista; Las siguientes funciones y métodos son parte del código que hace posible que funcione y corra adecuadamente el programa:

- void Menu(lista* tabla_hash)

Descripción: Da opciones al usuario de lo que desea hacer

Recibe: lista* tabla_hash

- void guardar(lista* tabla_hash, char* ruta)

Descripción: Guarda la información proporcionada por el usuario

Recibe: lista* tabla_hash, char* ruta

- void filtrar(lista* tabla_hash)

Descripción: Se pide información para la búsqueda y da opciones de lo que se desea hacer

Recibe: lista* tabla_hash

- void Estadisticas(lista * tabla_hash)

Descripción: este método permite mostrar las estadísticas del recorrido que se hizo para llegar a la palabra, definición, etc que el usuario deseaba.

Recibe: lista * tabla_hash

- int Hashing(char* palabra, int mostrar)

Descripción: Permite mapear la letra en un número, para una búsqueda de código más fácil

Recibe: char* palabra, int mostrar

Devuelve: regresar

- posicion Buscar(lista* tabla_hash, char* a_buscar, int mostrar)

Descripción: busca la clave obtenida del hash

Recibe: lista* tabla_hash, char* a_buscar, int mostrar

Devuelve: regresar

- void Eliminar_Palabra(lista* tabla_hash, char* palabra)

Descripción: el método elimina una palabra del diccionario

Recibe: lista* tabla_hash, char* palabra

- void Modificar_Definicion(lista* tabla_hash, char* palabra)

Descripción: El método permite modificar una definición

Recibe: lista* tabla_hash, char* palabra

- void Anadir_Palabra(lista* tabla_hash, elemento mi_elemento, int mostrar)

Descripción: permite añadir una palabra al diccionario

Recibe: lista* tabla_hash, elemento mi_elemento, int mostrar

- void Guardar_Diccionario(char* nombre, lista* tabla_hash)

Descripción: Permite guardar en el diccionario

Recibe: char* nombre, lista* tabla_hash

- void Guardar_Palabra(lista * tabla_hash, posicion mi_posicion, char* nombre)

Descripcion: permite guardar una palabra

Recibe: Guardar_Palabra(lista * tabla_hash, posicion mi_posicion, char* nombre)

- void Leer_Archivos(char* ruta, lista* tabla_hash)

Descripcion: permite leer los archivos que se incluyan

Recibe: char* ruta, lista* tabla_hash

- void Buscar_Letra(char letra, lista* tabla_hash)

Descripcion: Permite hacer una busqueda de la palabra que se introduzca

Recibe: char letra, lista* tabla_hash

- void Buscar_Sub_Cadena(lista* tabla_hash, char* subcadena)

Descripcion: busca una subcadena

Recibe: lista* tabla_hash, char* subcadena

Las funciones anteriores, solo son parte del código, pero son las básicas para que funcione correctamente el programa.

FUNCIONAMIENTO



Figura 1

En esta imagen se aprecia el menú, en donde el usuario podrá elegir lo que desea hacer.

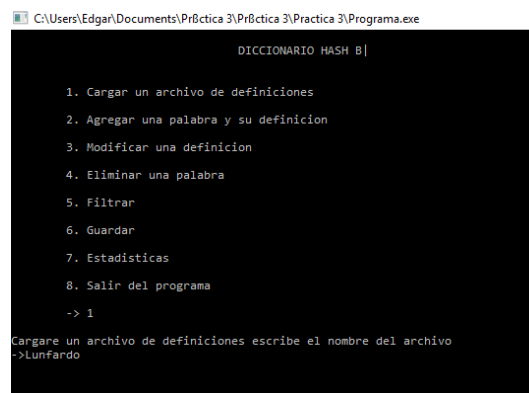


Figura 2

En esta prueba, el usuario eligió cargar un archivo, y coloco la palabra.

```

C:\Users\Edgar\Documents\Práctica 3\Práctica 3\Practica 3\Programa.exe
->La palabra Violero tiene un valor 38.975116 al hacer hashing
->La palabra Viorsi tiene un valor 35.316228 al hacer hashing
->La palabra Viscachazo tiene un valor 55.678738 al hacer hashing
->La palabra Viscachear tiene un valor 54.213508 al hacer hashing
->La palabra Vudua tiene un valor 31.857422 al hacer hashing
->La palabra Vivanco tiene un valor 38.975116 al hacer hashing
->La palabra Viviyo tiene un valor 33.407243 al hacer hashing
->La palabra Vivo tiene un valor 22.271495 al hacer hashing
->La palabra Viyuya tiene un valor 38.228907 al hacer hashing
->La palabra Viyuyera tiene un valor 50.971875 al hacer hashing
->La palabra Voracero tiene un valor 44.542990 al hacer hashing
->La palabra Votacen tiene un valor 39.329436 al hacer hashing
->La palabra Vovi tiene un valor 29.430190 al hacer hashing
->La palabra vovina tiene un valor 38.228907 al hacer hashing
->La palabra Vuelo tiene un valor 27.839369 al hacer hashing
->La palabra Yanta, en tiene un valor 50.566417 al hacer hashing
->La palabra Yeca tiene un valor 25.485938 al hacer hashing
->La palabra Veite tiene un valor 30.595742 al hacer hashing
->La palabra Veta tiene un valor 25.485938 al hacer hashing
->La palabra Veyatore tiene un valor 48.953187 al hacer hashing
->La palabra Vigolo tiene un valor 33.407243 al hacer hashing
->La palabra Vira tiene un valor 31.857422 al hacer hashing
->La palabra Yiranta tiene un valor 44.600391 al hacer hashing
->La palabra Yiradicta tiene un valor 57.343360 al hacer hashing
->La palabra Yirar tiene un valor 27.106754 al hacer hashing
->La palabra Viro tiene un valor 22.271495 al hacer hashing
->La palabra Yompa tiene un valor 31.857422 al hacer hashing
->La palabra Yoni tiene un valor 23.544152 al hacer hashing
->La palabra Yugaba tiene un valor 38.228907 al hacer hashing
->La palabra Yugador tiene un valor 43.370806 al hacer hashing
->La palabra yugante tiene un valor 48.953187 al hacer hashing
->La palabra Yugar tiene un valor 27.106754 al hacer hashing
->La palabra Yuguiyo tiene un valor 38.975116 al hacer hashing
->La palabra Yunta tiene un valor 31.857422 al hacer hashing
->La palabra Yurno tiene un valor 27.839369 al hacer hashing
->La palabra Yuta tiene un valor 25.485938 al hacer hashing
->La palabra Yuto tiene un valor 22.271495 al hacer hashing
->La palabra Zarzo tiene un valor 27.839369 al hacer hashing
->La palabra Zafar tiene un valor 27.106754 al hacer hashing
->La palabra Zampar tiene un valor 32.528105 al hacer hashing
->La palabra Zanagoria (Zanahoria) tiene un valor 316.553994 al hacer hashing
->La palabra Zapaba tiene un valor 38.228907 al hacer hashing
->La palabra Zapar tiene un valor 27.106754 al hacer hashing
->La palabra Zaparrastrozo tiene un valor 72.382359 al hacer hashing
->La palabra Zapayaso tiene un valor 44.542990 al hacer hashing
->La palabra Zapayero tiene un valor 44.542990 al hacer hashing
->La palabra Zapayo tiene un valor 33.407243 al hacer hashing
->La palabra Zaranda tiene un valor 44.600391 al hacer hashing
->La palabra Zarpap tiene un valor 32.528105 al hacer hashing

```

Figura 3

Se observa la clave que tiene cada palabra y se busca la que es similar a la de la palabra introducida.

```

DICCIONARIO HASH B|

1. Cargar un archivo de definiciones
2. Agregar una palabra y su definicion
3. Modificar una definicion
4. Eliminar una palabra
5. Filtrar
6. Guardar
7. Estadisticas
8. Salir del programa

-> 1

Cargare un archivo de definiciones escribe el nombre del archivo
->Glosario BIOQUANTUM

```

Figura 4

De nuevo se observa el menú, pero ahora se carga con un archivo distinto.

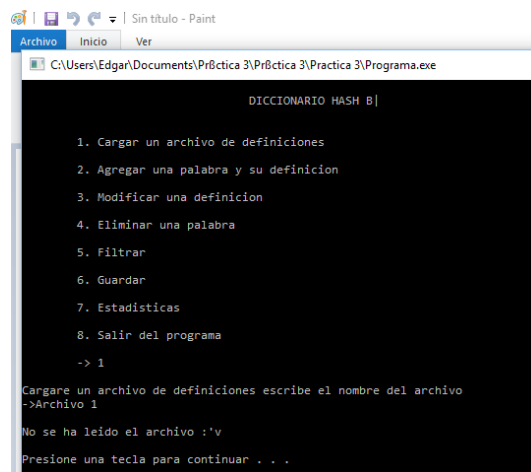
```

C:\Users\Edgar\Documents\Práctica 3\Práctica 3\Practica 3\Programa.exe
->La palabra Flagrancia tiene un valor 63.714844 al hacer hashing
->La palabra Flamante tiene un valor 48.953187 al hacer hashing
->La palabra Formato, dar tiene un valor 65.056209 al hacer hashing
->La palabra Frecuencia tiene un valor 63.714844 al hacer hashing
->La palabra Fructifero tiene un valor 55.678738 al hacer hashing
->La palabra Fuero tiene un valor 27.839369 al hacer hashing
->La palabra Galeno tiene un valor 33.407243 al hacer hashing
->La palabra Gelido tiene un valor 33.407243 al hacer hashing
->La palabra Geriatrico tiene un valor 55.678738 al hacer hashing
->La palabra Glosario tiene un valor 44.542990 al hacer hashing
->La palabra Habitat tiene un valor 37.295154 al hacer hashing
->La palabra Hacer conocido lo desconocido tiene un valor 161.468339 al hacer hashing
->La palabra Hallazgo tiene un valor 44.542990 al hacer hashing
->La palabra Hay tiene un valor 15.323157 al hacer hashing
->La palabra Hermetico tiene un valor 50.110864 al hacer hashing
->La palabra Histrion tiene un valor 44.947926 al hacer hashing
->La palabra Huraco tiene un valor 33.407243 al hacer hashing
->La palabra Ignorancia tiene un valor 63.714844 al hacer hashing
->La palabra Ignoto tiene un valor 33.407243 al hacer hashing
->La palabra Ilusion tiene un valor 39.329436 al hacer hashing
->La palabra Imperecedero tiene un valor 66.814485 al hacer hashing
->La palabra Inadmisibile tiene un valor 67.310632 al hacer hashing
->La palabra Inanidon tiene un valor 50.566417 al hacer hashing
->La palabra Incesante tiene un valor 55.072336 al hacer hashing
->La palabra Incolume tiene un valor 48.953187 al hacer hashing
->La palabra Inculcar tiene un valor 43.370806 al hacer hashing
->La palabra Indeleble tiene un valor 55.072336 al hacer hashing
->La palabra Indolente tiene un valor 55.072336 al hacer hashing
->La palabra Inducir tiene un valor 37.949455 al hacer hashing
->La palabra Infinito tiene un valor 44.542990 al hacer hashing
->La palabra Infligir tiene un valor 43.370806 al hacer hashing
->La palabra Ingrato tiene un valor 38.975116 al hacer hashing
->La palabra Innato tiene un valor 33.407243 al hacer hashing
->La palabra Inoculacion tiene un valor 61.803399 al hacer hashing
->La palabra Inscrito tiene un valor 44.542990 al hacer hashing
->La palabra Insipido tiene un valor 44.542990 al hacer hashing
->La palabra integro tiene un valor 38.975116 al hacer hashing
->La palabra Involuntario tiene un valor 66.814485 al hacer hashing
->La palabra Jurado tiene un valor 33.407243 al hacer hashing
->La palabra Jurar tiene un valor 27.106754 al hacer hashing
->La palabra Karma tiene un valor 31.857422 al hacer hashing
->La palabra Lacerar tiene un valor 37.949455 al hacer hashing
->La palabra Lanza tiene un valor 32.528105 al hacer hashing
->La palabra Limbo tiene un valor 38.975116 al hacer hashing
->La palabra Limbo tiene un valor 27.839369 al hacer hashing
->La palabra Limpido tiene un valor 38.975116 al hacer hashing
->La palabra Llano tiene un valor 27.839369 al hacer hashing
->La palabra Longevidad tiene un valor 61.803399 al hacer hashing
->La palabra Lozania tiene un valor 44.600391 al hacer hashing

```

Figura 5

Se observa nuevamente como hace el recorrido, la búsqueda para encontrar la clave similar.



```
Sin título - Paint
C:\Users\Edgar\Documents\Pr8ctica 3\Pr8ctica 3\Practica 3\Programa.exe

DICCIONARIO HASH B|

1. Cargar un archivo de definiciones
2. Agregar una palabra y su definicion
3. Modificar una definicion
4. Eliminar una palabra
5. Filtrar
6. Guardar
7. Estadisticas
8. Salir del programa
-> 1

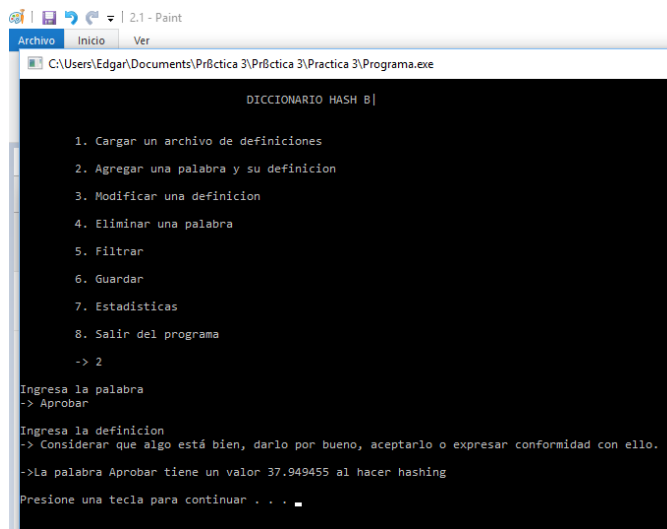
Cargare un archivo de definiciones escribe el nombre del archivo
-> Archivo 1

No se ha leído el archivo :\'v

Presione una tecla para continuar . . .
```

Figura 6

Se observa que ese archivo no se ha leído, debido a que no está incluido.



```
Sin título - Paint
C:\Users\Edgar\Documents\Pr8ctica 3\Pr8ctica 3\Practica 3\Programa.exe

DICCIONARIO HASH B|

1. Cargar un archivo de definiciones
2. Agregar una palabra y su definicion
3. Modificar una definicion
4. Eliminar una palabra
5. Filtrar
6. Guardar
7. Estadisticas
8. Salir del programa
-> 2

Ingresa la palabra
-> Aprobar

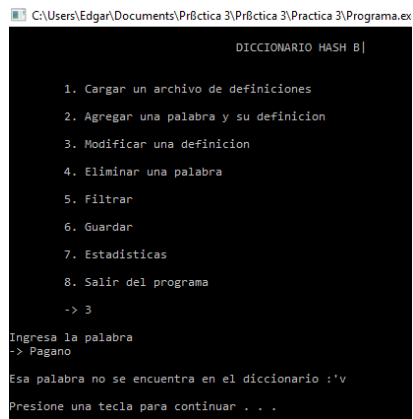
Ingresa la definicion
-> Considerar que algo está bien, darlo por bueno, aceptarlo o expresar conformidad con ello.

-> La palabra Aprobar tiene un valor 37.949455 al hacer hashing

Presione una tecla para continuar . . .
```

Figura 7

Aquí se ha buscado correctamente la palabra y se ha dado su definición.



```
C:\Users\Edgar\Documents\Pr8ctica 3\Pr8ctica 3\Practica 3\Programa.exe

DICCIONARIO HASH B|

1. Cargar un archivo de definiciones
2. Agregar una palabra y su definicion
3. Modificar una definicion
4. Eliminar una palabra
5. Filtrar
6. Guardar
7. Estadisticas
8. Salir del programa
-> 3

Ingresa la palabra
-> Pagano

Esa palabra no se encuentra en el diccionario :\'v

Presione una tecla para continuar . . .
```

Figura 8

Ahora bien, aquí indica que esa palabra no se encuentra incluida en el diccionario.


```

C:\Users\Edgar\Documents\Pr8ctica 3\Pr8ctica 3\Practica 3\Programa.exe

DICCIONARIO HASH B|

1. Cargar un archivo de definiciones
2. Agregar una palabra y su definicion
3. Modificar una definicion
4. Eliminar una palabra
5. Filtrar
6. Guardar
7. Estadisticas
8. Salir del programa
-> 4

Que palabra sera eliminada?
-> Jurar

->La palabra Jurar tiene un valor 27.186754 al hacer hashing
->Di 11 pasos para encontrar la palabra o verificar si existe Jurar
Se ha eliminado la palabra Jurar de manera satisfactoria :D
Presione una tecla para continuar . . .

```

Figura 9

En esta imagen se aprecia la palabra que se desea eliminar. Y se observa que efectivamente ha sido eliminada.

```

C:\Users\Edgar\Documents\Pr8ctica 3\Pr8ctica 3\Practica 3\Programa.exe

DICCIONARIO HASH B|

1. Cargar un archivo de definiciones
2. Agregar una palabra y su definicion
3. Modificar una definicion
4. Eliminar una palabra
5. Filtrar
6. Guardar
7. Estadisticas
8. Salir del programa
-> 5

Filtrar diccionario
1. Buscar por sub-cadena
2. Buscar por Letra de inicio
3. Buscar palabra
->

```

Figura 10

En esta parte, se elige la opción 5, la cual es filtrar, en ella vemos que proporciona otro menú.

```

C:\Users\Edgar\Documents\Pr8ctica 3\Pr8ctica 3\Practica 3\Programa.exe

1. Cargar un archivo de definiciones
2. Agregar una palabra y su definicion
3. Modificar una definicion
4. Eliminar una palabra
5. Filtrar
6. Guardar
7. Estadisticas
8. Salir del programa
-> 5

Filtrar diccionario
1. Buscar por sub-cadena
2. Buscar por Letra de inicio
3. Buscar palabra
-> 1

Que sub-cadena buscare?
-> Men

1. Mente: Es el producto de la acción de los flujos de conciencia y energía en el cerebro que crea formas de pensamiento, segmentos holograficos o patrones neurosinapticos llamados memoria. La conciencia y energía son lo que mantienen vivo al cerebro; son su fuente de poder. La capacidad de pensar de una persona es lo que la provee de una mente.
-> -> di 341 pasos para encontrar esta palabra

2. Menta: Fama, voz u opinion acerca de un persona.
-> -> di 418 pasos para encontrar esta palabra

3. Menega: Dinero
-> -> di 966 pasos para encontrar esta palabra

4. Menester: Profesion, ministerio, empleo.
-> -> di 1246 pasos para encontrar esta palabra

5. Menesunda: Droga / Desorden, lio.
-> -> di 2222 pasos para encontrar esta palabra

6. Meneguina: Dinero.
-> -> di 2223 pasos para encontrar esta palabra

```

Figura 11. Se ve que la palabra introducida, tiene diversos resultados.

```

C:\Users\Edgar\Documents\Práctica 3\Práctica 3\Practica 3\Programa.exe
Filtrar diccionario
1. Buscar por sub-cadena
2. Buscar por letra de inicio
3. Buscar palabra
-> 2
    Que letra_buscare?
-> Z
1. Zurdo: el corazon.-> ->di 203 saltos
2. Zaper: Trabajar esforzadamente.
-> ->di 204 saltos
3. Zafar: Escapar, librarse.
-> ->di 205 saltos
4. Zarzo: Anillo
-> ->di 206 saltos
5. Zarpas: Irse rapido.
-> ->di 479 saltos
6. Zampar: Enjaretar / Encajar.
-> ->di 480 saltos
7. Zapayo: Cabeza.
-> ->di 571 saltos
8. Zapaba: Trabajaba.
-> ->di 883 saltos
9. Zaranda: Zurra.
-> ->di 1369 saltos
10. Zapayero: Desordenado.
-> ->di 1370 saltos
11. Zapayaso: Azar, imprevisto / Punetazo.
-> ->di 1371 saltos
12. Zumbarse: Drogarse.
-> ->di 1673 saltos
13. Zaparastrozo: Andrajoso.
-> ->di 2527 saltos
14. Zanagoria (Zanahoria): Tonto.
-> ->di 2899 saltos

```

Figura 12. Al igual que la anterior la letra tiene diversos resultados.

```

C:\Users\Edgar\Documents\Práctica 3\Práctica 3\Practica 3\Programa.exe
DICCIONARIO HASH B|
1. Cargar un archivo de definiciones
2. Agregar una palabra y su definicion
3. Modificar una definicion
4. Eliminar una palabra
5. Filtrar
6. Guardar
7. Estadisticas
8. Salir del programa
-> 5
Filtrar diccionario
1. Buscar por sub-cadena
2. Buscar por letra de inicio
3. Buscar palabra
-> 3
    Que palabra buscare?
-> Protagonista
->La palabra Protagonista tiene un valor 76.457813 al hacer hashing
    ->Di 3 pasos para encontrar la palabra o verificar si existe Protagonista
    Existe la palabra Protagonista su definicion es: Que tiene la parte principal.
Presione una tecla para continuar . . .

```

Figura 13

Al introducir la palabra, se encontró solo un resultado, debido a que solo existía una clave similar a la de ella.

```

DICCIONARIO HASH B|
1. Cargar un archivo de definiciones
2. Agregar una palabra y su definicion
3. Modificar una definicion
4. Eliminar una palabra
5. Filtrar
6. Guardar
7. Estadisticas
8. Salir del programa
-> 6
    Como se llamara el archivo?
-> Archivo de prueba 1
    Elige una opcion
    1.Exportar todo el diccionario
    2.Exportar una palabra
    -> 1
Presione una tecla para continuar . . .

```

Figura 14

Aquí observamos que se guardara un archivo, y de nuevo, ofrecen dos opciones al usuario. Se elige la uno, por lo tanto e exporta el archivo que desee.

```

C:\Users\Edgar\Documents\Práctica 3\Práctica 3\Practica 3\Programa.exe
DICCIONARIO HASH B|

1. Cargar un archivo de definiciones
2. Agregar una palabra y su definicion
3. Modificar una definicion
4. Eliminar una palabra
5. Filtrar
6. Guardar
7. Estadísticas
8. Salir del programa
-> 6

Como se llamara el archivo?
-> Archivo de prueba 2

Elige una opcion
1.Exportar todo el diccionario
2.Exportar una palabra
-> 2

Que palabra sera exportada?
->Exento

->La palabra Exento tiene un valor 33.407243 al hacer hashing
->Di 29 pasos para encontrar la palabra o verificar si existe Exento
Presione una tecla para continuar . . .

```

Figura 15

A diferencia de la anterior, aquí solo se exportar una palabra y efectivamente lo hace perfectamente.

```

C:\Users\Edgar\Documents\Práctica 3\Práctica 3\Practica 3\Programa.exe
DICCIONARIO HASH B|

1. Cargar un archivo de definiciones
2. Agregar una palabra y su definicion
3. Modificar una definicion
4. Eliminar una palabra
5. Filtrar
6. Guardar
7. Estadísticas
8. Salir del programa
-> 7

El orden máximo de busquedas es 0²
El tamaño de nuestra tabla hash es de 90 actualmente
La tabla hash tiene un promedio de 28.677778
Presione una tecla para continuar . . .

```

Figura 16

Esta imagen nos presenta las estadísticas que se tienen del recorrido que realizó el programa para lograr el pedido del usuario.

```

C:\Users\Edgar\Documents\Práctica 3\Práctica 3\Practica 3\Programa.exe
DICCIONARIO HASH B|

1. Cargar un archivo de definiciones
2. Agregar una palabra y su definicion
3. Modificar una definicion
4. Eliminar una palabra
5. Filtrar
6. Guardar
7. Estadísticas
8. Salir del programa
-> 8

.....
Process exited after 1577 seconds with return value 0
Presione una tecla para continuar . . .

```

Figura 17

Finalmente se aprecia en la opción 8, la salida o cierre del programa.

ERRORES DETECTADOS

El programa permite poder agregar nuevas palabras, consulta el significado de alguna, modifica el significado y elimina palabras del diccionario hash. Adema es capaz de cargar un archivo de palabras y sus definiciones.

El usuario puede exportar en determinado momento la lista de palabras a un archivo.

Se pueden buscar todas las palabras que inicien con: letra, frase o que contengan una subcadena.

Es decir, el programa realizado, cumple con los estándares de funcionamiento para el usuario. Es decir, el programa no tuvo error alguno; todo se cumplió perfectamente y se mostró correctamente a pantalla.

POSIBLES MEJORAS

La única mejora que se podría hacer en la implementación, es cambiar la forma en que se detecta el fin y el inicio del significado de una palabra o bien cadena.

CONCLUSIONES

Núñez García Tania Itzel

Gracias a la práctica realizada y reportada en este documento, es posible concluir que el uso de tablas hash es una técnica de gran ayuda para utilizar como recurso para programas en los que se necesita realizar las operaciones de inserción, eliminación y búsqueda en tiempo constante.

Con esta práctica fue posible comprobar el funcionamiento del hashing, puesto que se utiliza una gran cantidad de información a guardar y, al querer acceder a algo, el proceso se hace muy lento; para esto sirve la función de hash: para poder acceder rápidamente a estos.

La tabla hash toma en cuenta solamente un valor absoluto, interpretado como un valor numérico; este se le asigna mediante una buena programación de la función hash, para que no existan "colisiones"; al momento de asignar un valor para cada dato, esto es, existe la posibilidad de que se obtenga el mismo número de asignación.

Y aquí, considero que recayó realmente la dificultad de la práctica, en poder programar apropiadamente la función con el fin de hacer lo más mínima posible la probabilidad de colisiones; aquí es donde también utilizamos algo previamente visto: las listas.

Por último, considero que es importante recordar bien toda la parte teórica también, para poder saber, en un futuro, el método más apropiado para crear un programa.

Quiros Díaz Verónica Jackeline

Esta práctica permitió concluir que el uso del Hash es una de las herramientas más utilizadas para programar y procesar más eficientemente sistemas que usamos día tras día, ya que permite agilizar operaciones y dar solución a procesos que ameritan un largo tiempo para desarrollarse. Es decir, una tabla hash suele tener como principal ventaja el acceso a los datos muy rápidamente, si se cumple que la función este bien diseñada para que no se produzcan demasiadas colisiones, o sea, la principal operación que realiza de manera eficiente, es la búsqueda, ya que permite el acceso a los elementos almacenados a partir de una clave de este. Lo interesante es que no recorre los elementos sino que checa al elemento, le aplica la función de hash para obtener un número, con el accede a la posición de la lista y luego compara con el elemento que contiene esa posición.

Además hay que recalcar que la lista creada previamente en clase, es de gran utilidad ya que es genérica, o bien, permite hacer uso de ella en cualquier problema que se presente.

Vargas Romero Erick Efraín

En esta práctica nos hemos percatado de lo eficiente que pueden ser las tablas hash en algún programa en el que tengamos varios elementos que requiramos ordenar, pero el pilar del uso de las tablas hash es la función de hashing, la cual por así decirlo asigna una especie de ID a algún elemento en nuestro caso el hashing fue hecho utilizando una palabra, y realizando otras operaciones aritméticas que nos permiten tener una buena distribución de los elementos en nuestra tabla hash. También hicimos uso de las listas doblemente ligadas y son de gran utilidad, en este caso cada "parte" de la tabla hash tenía una lista doblemente ligada, ya que en concreto la tabla hash es un arreglo de listas.

También en esta práctica el manejo de archivos es o fue primordial, ya que es necesario que el usuario pueda importar palabras de un archivo de texto a este programa y también debe ser o mejor dicho es posible exportar palabras que ya están en el programa hacia un archivo de texto, ya sea exportado solo una palabra o todo el diccionario.

El manejo de la librería Sting también fue importante porque fue necesario realizar numerosas comparaciones con cadenas de carácter o bien buscar subcadena, cosa que es sencilla de hacer ya sea recorriendo el arreglo de caracteres o bien usando strstr para buscar subcadena.


```

case 2: //printf("Lalala 2 \n\n");
    fflush(stdin);
    printf("\nIngresa la palabra \n-> "); //se imprime a pantalla
    gets(mi_elemento.p); // almacena informacion
    fflush(stdin); //elimina la basura
    printf("\nIngresa la definicion \n-> "); //se imprime a pantalla
    gets(mi_elemento.d); //almacena
    Anadir_Palabra(tabla_hash, mi_elemento, 0);
    printf("\n\n"); //se imprime a pantalla
    system("PAUSE"); //termina el proceso
    system("CLS"); //limpia pantalla
    Menu(tabla_hash); //manda a llamara la funcion
    break;

case 3: //printf("Lalala 3 \n\n");
    fflush(stdin); //elimina basura
    printf("\nIngresa la palabra \n-> "); //se imprime a pantalla
    gets(mi_elemento.p); //almacena informacion
    Modificar_Definicion(tabla_hash, mi_elemento.p);
    printf("\n\n"); //se imprime a pantalla
    system("PAUSE"); // termina el proceso
    system("CLS"); //limpia a pantalla
    Menu(tabla_hash); //llama la funcion
    break;

case 4: //printf("Lalala 4 \n\n");
    fflush(stdin);
    printf("\nQue palabra sera eliminada? \n -> "); //se imprime a pantalla
    gets(mi_elemento.p);
    Eliminar_Palabra(tabla_hash, mi_elemento.p);
    printf("\n\n"); //se imprime a pantalla
    system("PAUSE");
    system("CLS"); //limpia la pantalla
    Menu(tabla_hash); //llama la funcion
    break;

case 5:
    printf("\n\tFiltrar diccionario"); //se imprime a pantalla
    filtrar(tabla_hash); //elimina la basura
    printf("\n\n"); //se imprime a pantalla
    system("PAUSE");
    system("CLS"); // limpia la pantalla
    Menu(tabla_hash); //llama la funcion
    break;

case 6:
    fflush(stdin); //elimina la basura
    printf("\n\tComo se llamara el archivo? \n -> "); //se imprime a pantalla
    gets(ruta); // almacena informacion
    guardar(tabla_hash, ruta); // llama la funcion

    printf("\n\n"); //se imprime a pantalla
    system("PAUSE");
    system("CLS"); //limpia la basura
    Menu(tabla_hash); // manda a llamara la funcion

case 7:
    Estadisticas(tabla_hash);
    printf("\n\n"); //se imprime a pantalla
    system("PAUSE");
    system("CLS"); //limpia la pantalla
    Menu(tabla_hash); //llama la funcion
    break;

case 8:

```

```

        exit (0); //termina el proceso
        break;

default:
    printf("\n\tEsta opcion no existe"); //se imprime a pantalla
    break;
}
}
/*void guardar(lista* tabla_hash, char* ruta)
Descripcion: Guarda la informacion proporcionada por el usuario
Recibe: lista* tabla_hash, char* ruta
*/

void guardar(lista* tabla_hash, char* ruta)
{
    int x; //se declara x
    posicion mi_posicion; //se declara mi_posicion
    char texto[100] = ""; // se crea el arreglo de 100
    printf("\n\tElige una opcion"); //se imprime a pantalla
    printf("\n\t1.Exportar todo el diccionario"); //se imprime a pantalla
    printf("\n\t2.Exportar una palabra \n\t -> "); //se imprime a pantalla
    scanf("%d", &x); //almacena informacion
    switch(x)
    {
        case 1:
            Guardar_Diccionario(ruta, tabla_hash); //llama la funcion
            break;
        case 2:
            fflush(stdin);
            printf("\n\tQue palabra sera exportada? \n\t ->"); //se imprime a pantalla
            gets(texto);
            mi_posicion = Buscar(tabla_hash, texto, 0);
            if(mi_posicion != NULL) //si mi_posicion no es NULL se realizara lo
siguiente
                Guardar_Palabra(tabla_hash, mi_posicion, ruta);
            break;
        default:
            printf("\n\tEsta opcion no existe"); //se imprime a pantalla
            break;
    }
}
}
/*
void filtrar(lista* tabla_hash)
Descripcion: Se pide informacion para la busqueda y da opciones de lo que se desea
hacer
Recibe: lista* tabla_hash
*/
void filtrar(lista* tabla_hash)
{
    int option; //se declara option
    char texto[100] = ""; char character;
    posicion mi_posicion = NULL; // se declara mi_posicion
    printf("\n\t1. Buscar por sub-cadena"); //se imprime a pantalla
    printf("\n\t2. Buscar por Letra de inicio"); //se imprime a pantalla
    printf("\n\t3. Buscar palabra \n-> ");
    scanf("%d", &option);
    switch(option)
    {
        case 1:
            fflush(stdin); // elimna la basura
            printf("\n\tQue sub-cadena buscare? \n -> "); //se imprime a pantalla

```



```

        gets(texto); //se almacena la informacion
        Buscar_Sub_Cadena(tabla_hash, texto); //se llama a la funcion
        //Menu(tabla_hash);
        break;
    case 2:
        fflush(stdin); // se elimina la basura
        printf("\n\tQue letra buscar? \n -> "); //se imprime a pantalla
        scanf("%c", &caracter); //se almacena la informacion
        Buscar_Letra(caracter, tabla_hash); //se llama a la funcion
        break;
    case 3:
        fflush(stdin); //se elimina la basura
        printf("\n\tQue palabra buscar? \n -> "); //se imprime a pantalla
        gets(texto); //se almacena la informacion
        mi_posicion = Buscar(tabla_hash, texto, 0);
        if(mi_posicion != NULL) // si se cumple que no es NULL , se imprimira lo
siguiente
            printf("\n\tExiste la palabra %s su definicion es: %s", mi_posicion-
>e.p, mi_posicion->e.d); //se imprime a pantalla
        else //de lo contrario
            printf("\n\tLa palabra %s no esta en el diccionario", texto); //se
imprime a pantalla esto
        break;
    default:
        printf("\n\tEsa opcion no existe"); // imprime a pantalla
        break;
    }
    //Menu(tabla_hash);
}

/*
void Estadisticas(lista * tabla_hash)
Descripcion:este metodo permite mostrar las estadisticas del recorrido que se hizo
para llegar a la palabra, definicion,
etc que el usuario deseaba.
Recibe: lista * tabla_hash
*/
void Estadisticas(lista * tabla_hash)
{
    int i, j, tam = 0, Colisiones = 0; //se declaran i,j, tam que se inicializa en
cero y colisiones que tambien se inicializa en cero
    for(i=0; i<1000; i++) // se termina el for, cuando se ya no se cumpla que i es
menor que 1000
    {
        if(!Empty(&tabla_hash[i])) // si se cumple que no esta vacia, se realizara
lo siguiente
        {
            tam++; //se incrementara tam
            Colisiones += Size(&tabla_hash[i]);
        }
    }
    printf("\n\tEl orden maximo de busquedas es 0%c", 253); //se imprime a pantalla
    printf("\n\tEl tamaño de nuestra tabla hash es de %d actualmente", tam); //se
imprime a pantalla
    printf("\n\tLa tabla hash tiene un promedio de %f",
(float)Colisiones/(float)tam); //se imprime a pantalla
}

```

BIBLIOGRAFÍA

http://ecampus.fca.unam.mx/ebook/imprimibles/informatica/informatica_3/Unidad_3.pdf

<http://blog.martincruz.me/2013/03/metodo-de-dispersion-hashing-en-c.html>

<http://codereview.stackexchange.com/questions/85556/simple-string-hashing-algorithm-implementation>

https://es.wikipedia.org/wiki/Tabla_hash#Ventajas_e_inconvenientes_de_las_tablas_hash