****

**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](#_Toc465014334)

[PLANTEAMIENTO DEL PROBLEMA 3](#_Toc465014335)

[DISEÑO Y FUNCIONAMIENTO DE LA SOLUCIÓN 4](#_Toc465014336)

[IMPLEMENTACIÓN DE LA SOLUCIÓN 5](#_Toc465014337)

[FUNCIONAMIENTO 6](#_Toc465014338)

[ERRORES DETECTADOS 12](#_Toc465014339)

[POSIBLES MEJORAS 12](#_Toc465014340)

[CONCLUSIONES 12](#_Toc465014341)

[ANEXO 14](#_Toc465014342)

[BIBLIOGRAFÍA 18](#_Toc465014343)

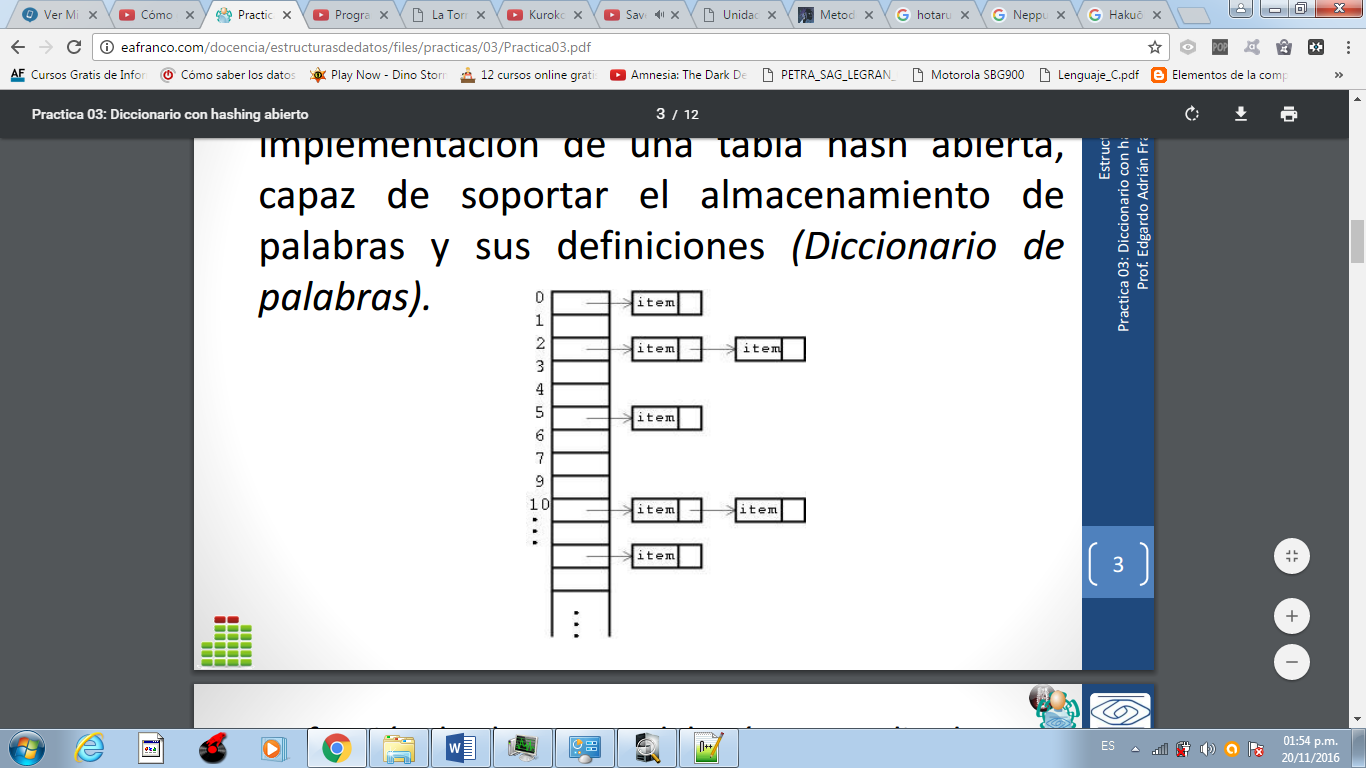
INTRODUCCIÓN

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.

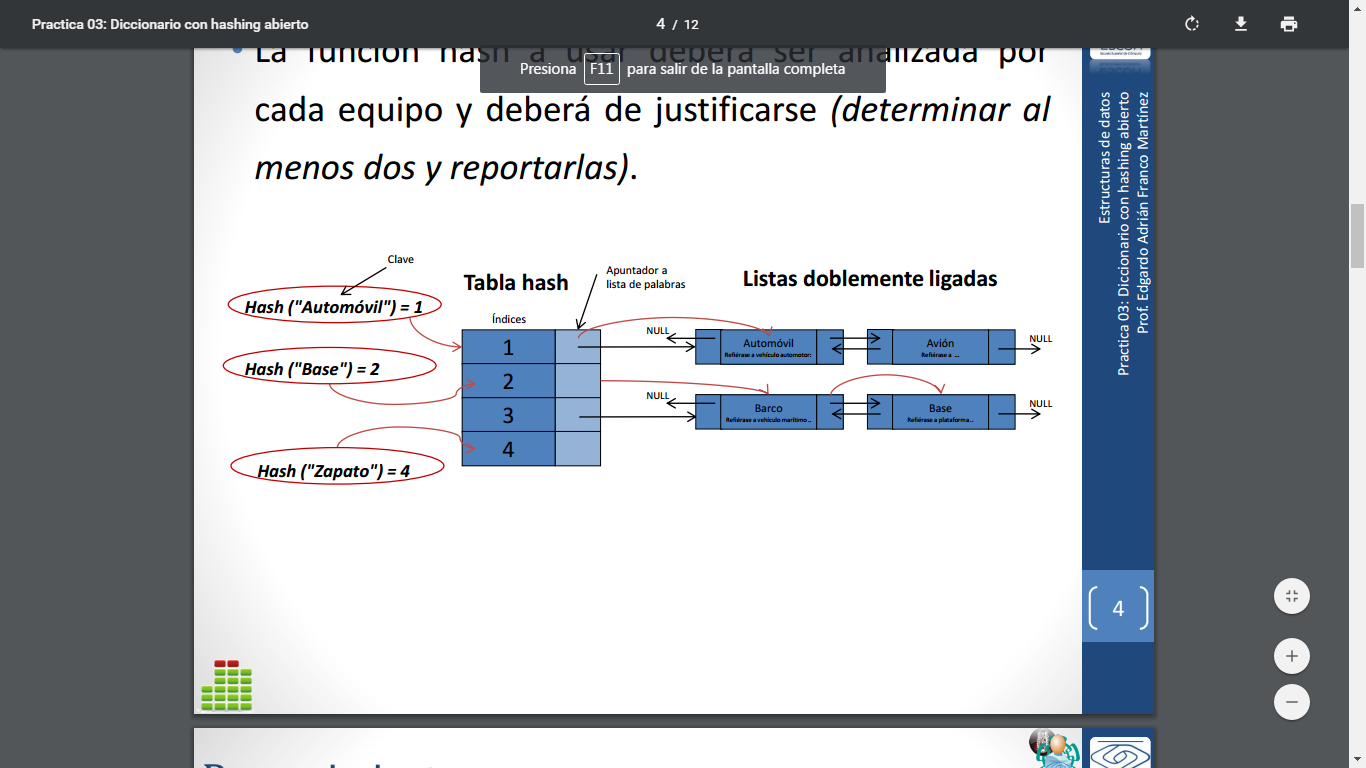
Existe el grupo de búsquedas por transformación de llaves (Hash), que aumenta la eficiencia, en cuanto al tiempo de ejecución, ya que accede a los registros por lo general más rápidamente, pero va a depender de su implementación. Dicho esto, se elaboró la práctica y presente reporte, para comprender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

Descripcion:Da opciones al usuario de lo que desea hacer

Recibe: lista\* tabla\_hash

* void guardar(lista\* tabla\_hash, char\* ruta)

Descripcion: Guarda la informacion proporcionadapor el usuario

Recibe: lista\* tabla\_hash, char\* ruta

* 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 Estadisticas(lista \* tabla\_hash)

Descripcion:este metodo permite mostrar las estadisticas del recorrido que se hizo para llegar a la palbra, definicion,etc que el usuario deseaba.

Recibe: lista \* tabla\_hash

* int Hashing(char\* palabra, int mostrar)

Descripcion: Permite mapear la letra en un número, para una busqueda de codigo mas facil

Recibe:char\* palabra, int mostrar

Devuelve: regresare

* posicion Buscar(lista\* tabla\_hash, char\* a\_buscar, int mostrar)

Descripcion:busca la clave obtenida del hasheo

Recibe:lista\* tabla\_hash, char\* a\_buscar, int mostrar

Devuelve: regresare

* void Eliminar\_Palabra(lista\* tabla\_hash, char\* palabra)

Descripcion: el metodo elimina una palabra del diccionario

Recibe:lista\* tabla\_hash, char\* palabra

* void Modificar\_Definicion(lista\* tabla\_hash, char\* palabra)

Descripcion: El metodo permite modificar una definicion

Recibe:lista\* tabla\_hash, char\* palabra

* void Anadir\_Palabra(lista\* tabla\_hash, elemento mi\_elemento, int mostrar)

Descripcion: permite añadir una palabra al diccionario

Recibe:lista\* tabla\_hash, elemento mi\_elemento, int mostrar

* void Guardar\_Diccionario(char\* nombre, lista\* tabla\_hash)

Descripcion: 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

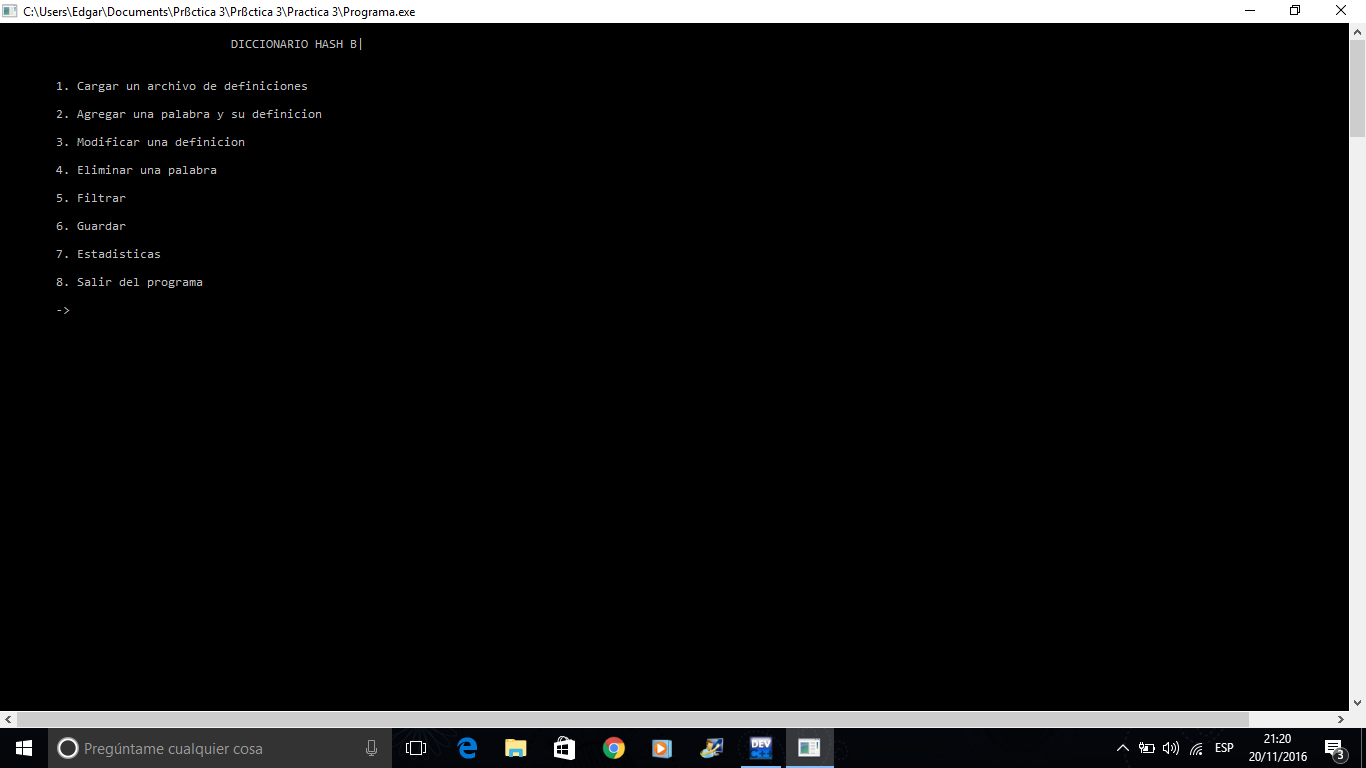
* void Buscar\_Sub\_Cadena(lista\* tabla\_hash, char\* subcadena)

Descripcion: busca una subcadena

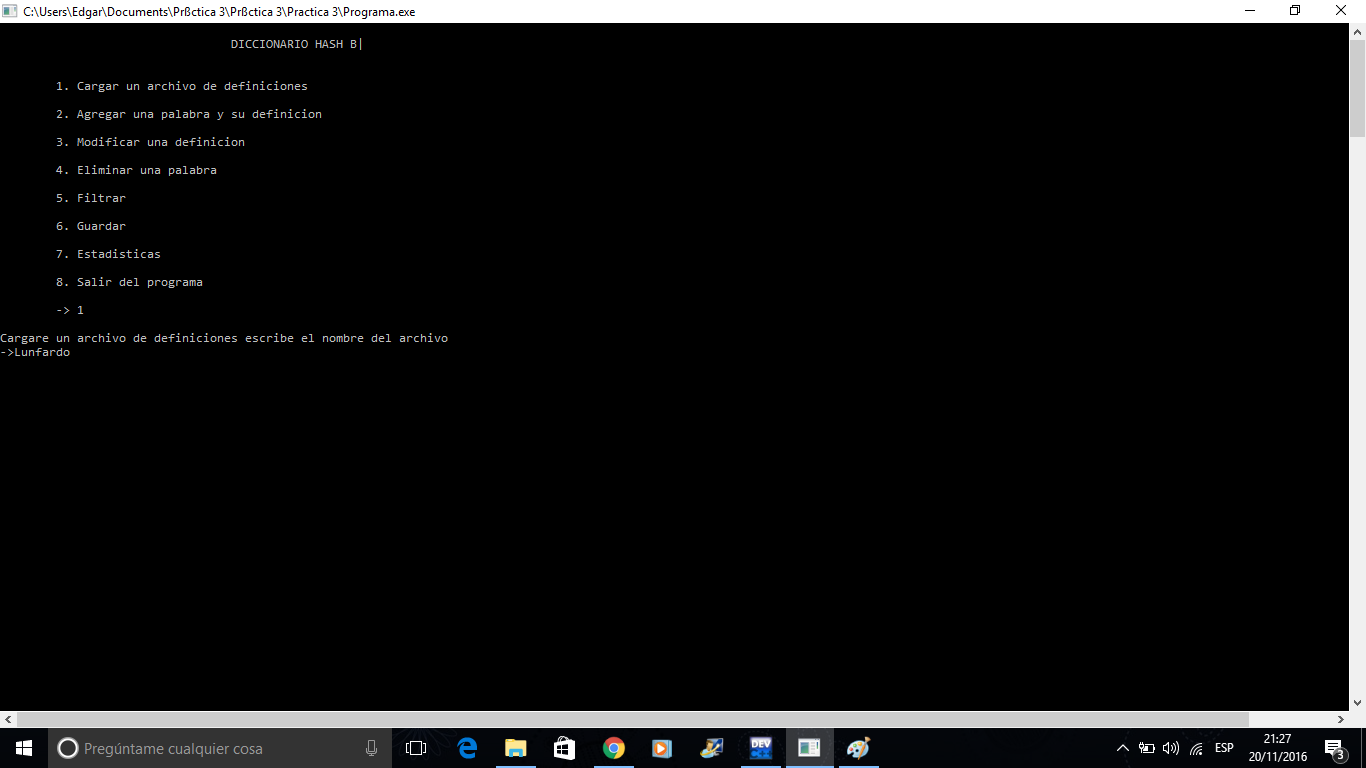
Recibe:lista\* tabla\_hash, char\* subcadena

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

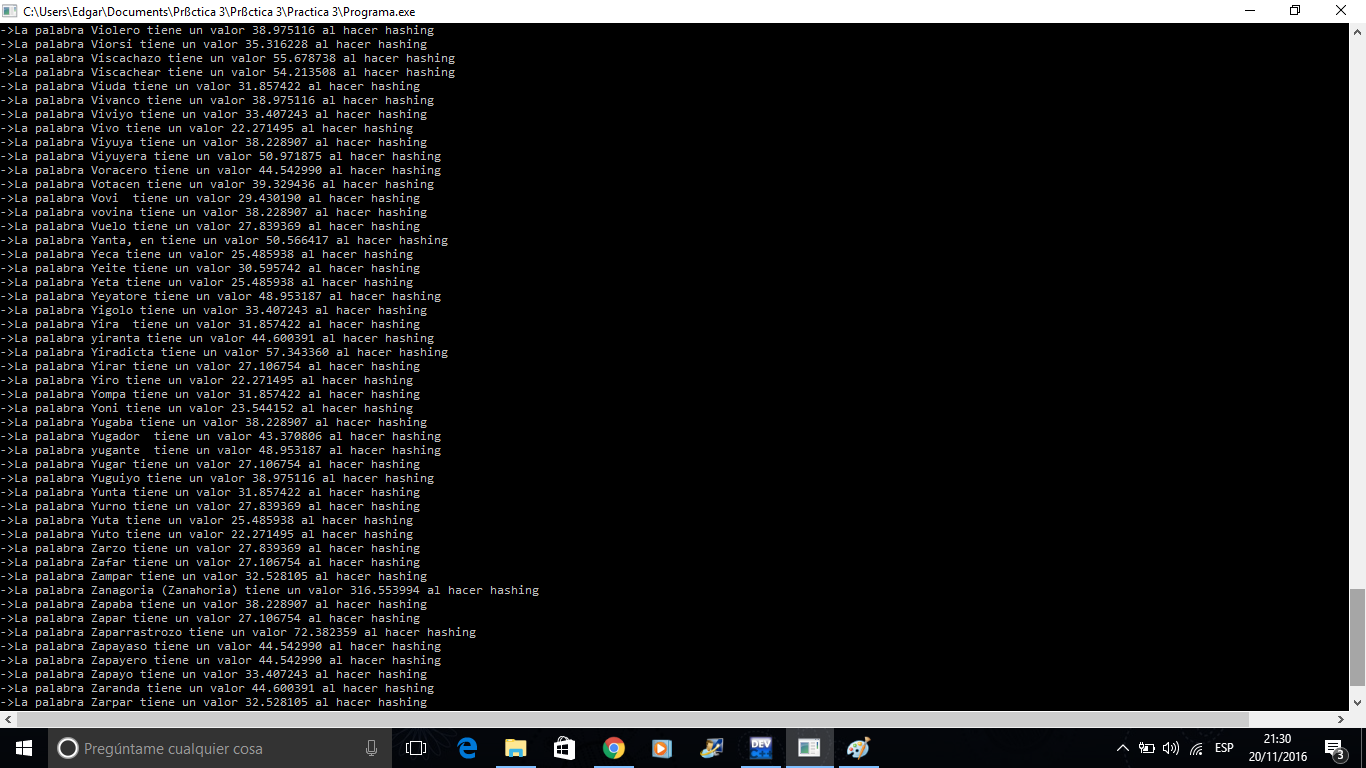
FUNCIONAMIENTO

Figura 1

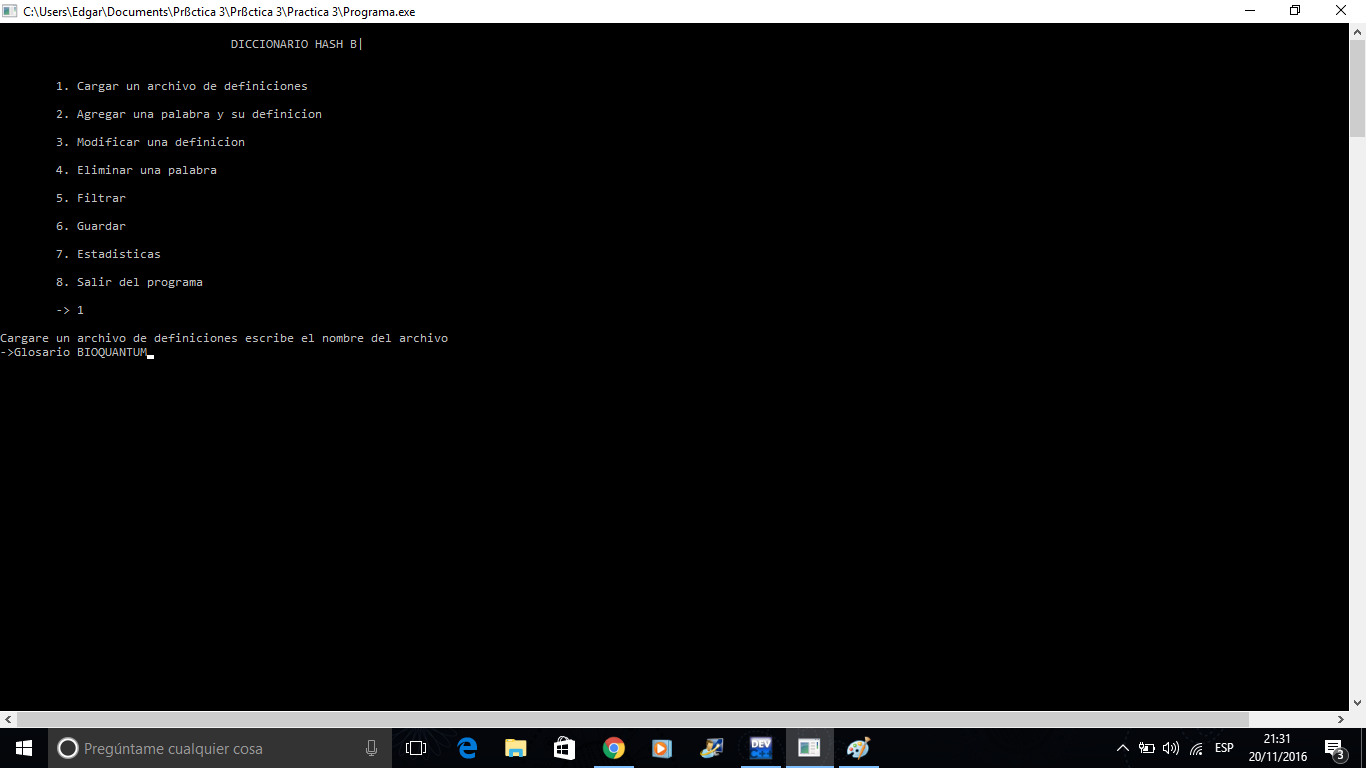
En esta imagen se aprecia el menú, en donde el usuario podrá elegir lo que desea hacer.

 Figura 2

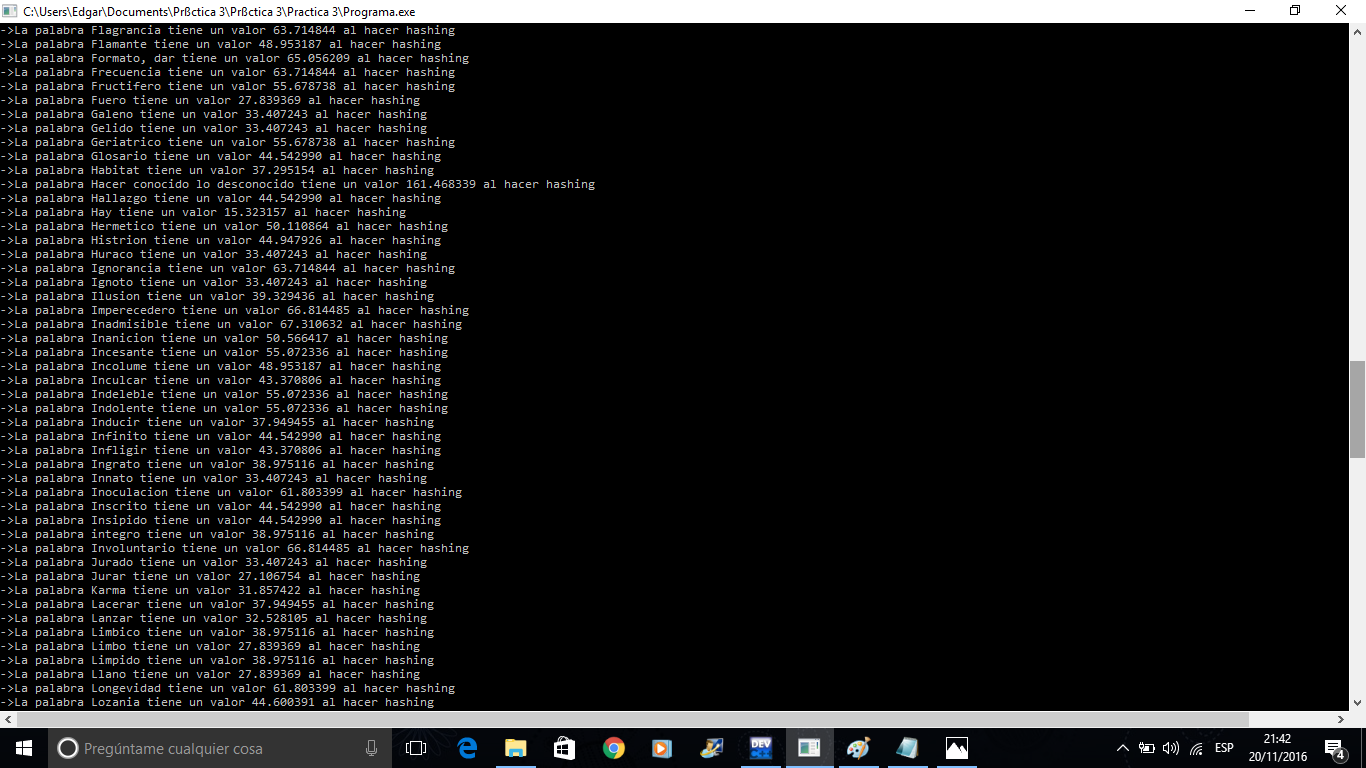
En esta prueba, el usuario eligió cargar un archivo, y coloco la palabra.

 Figura 3

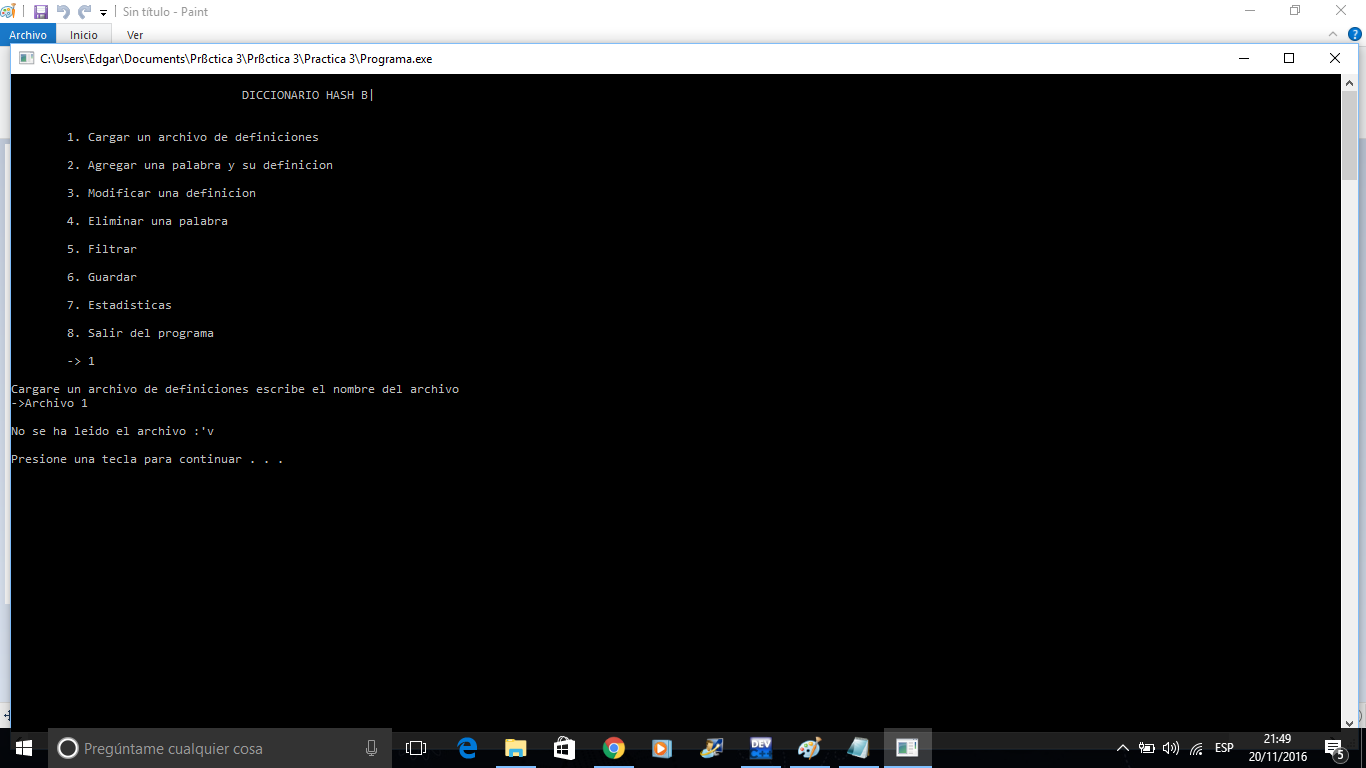
Se observa la clave que tiene cada palabra y se busca la que es similar a la de la palabra introducida.

 Figura 4

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

 Figura 5

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

 Figura 6

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

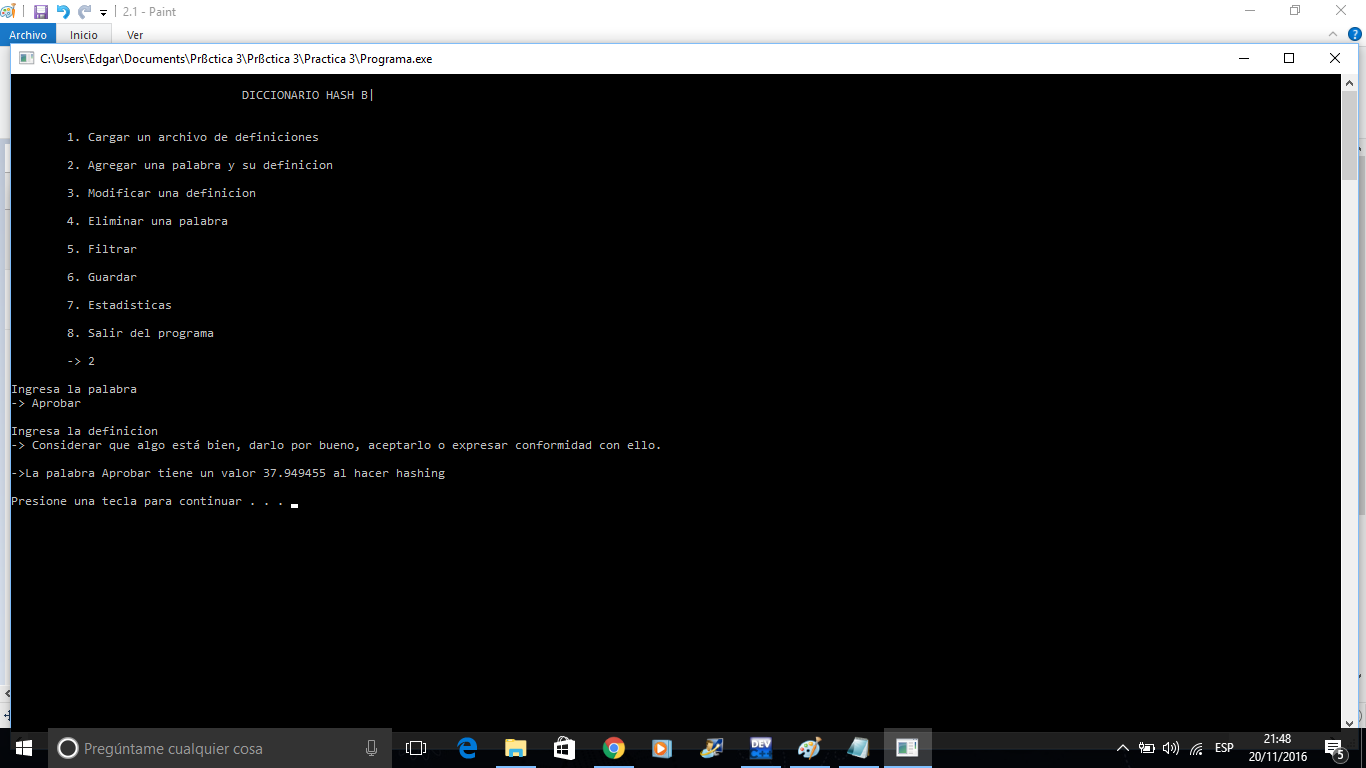
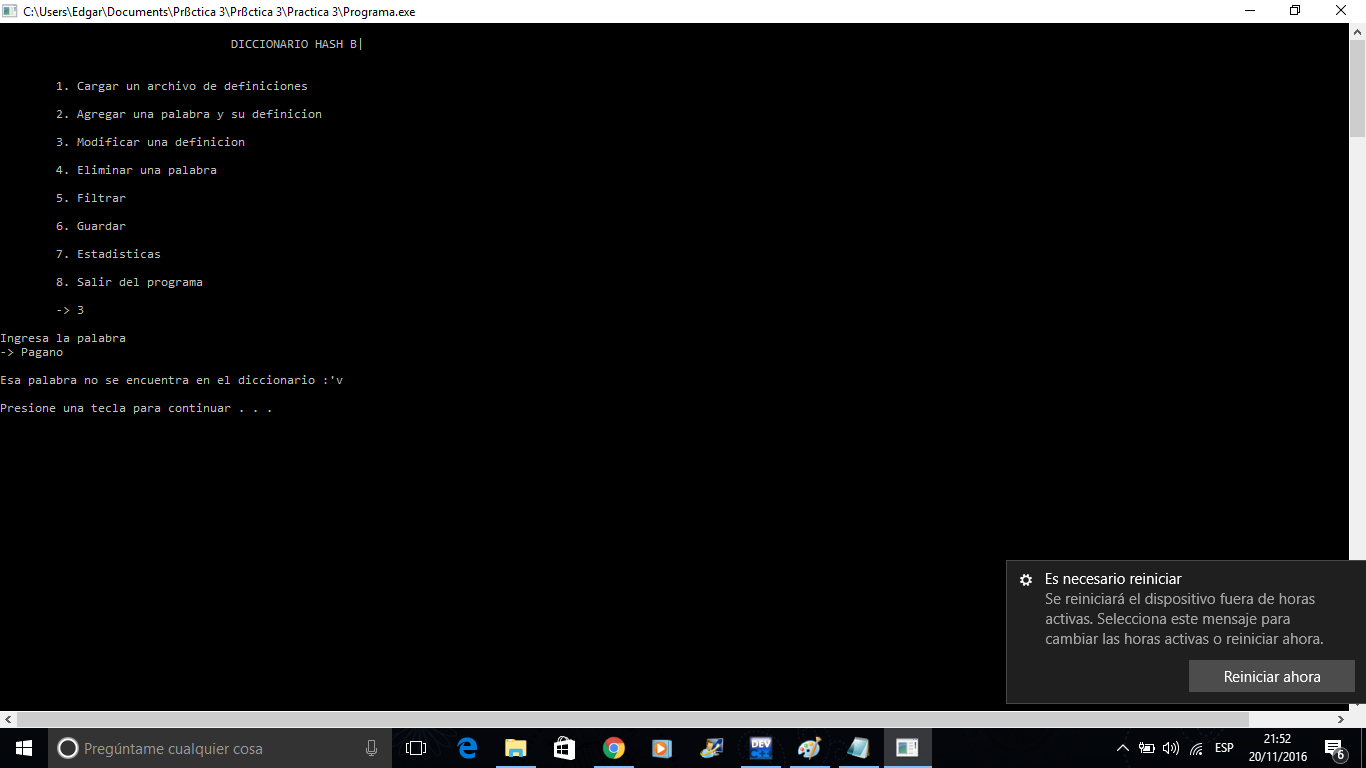
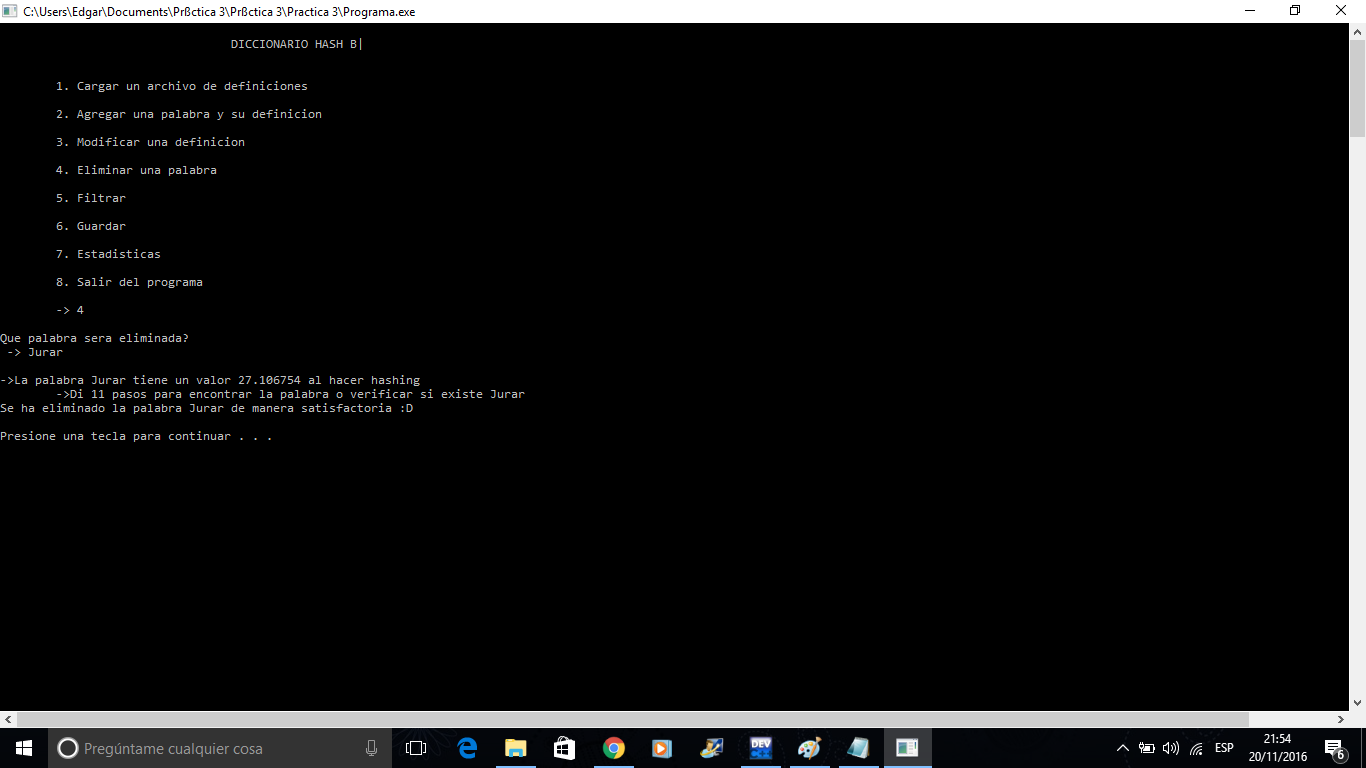


Figura 7

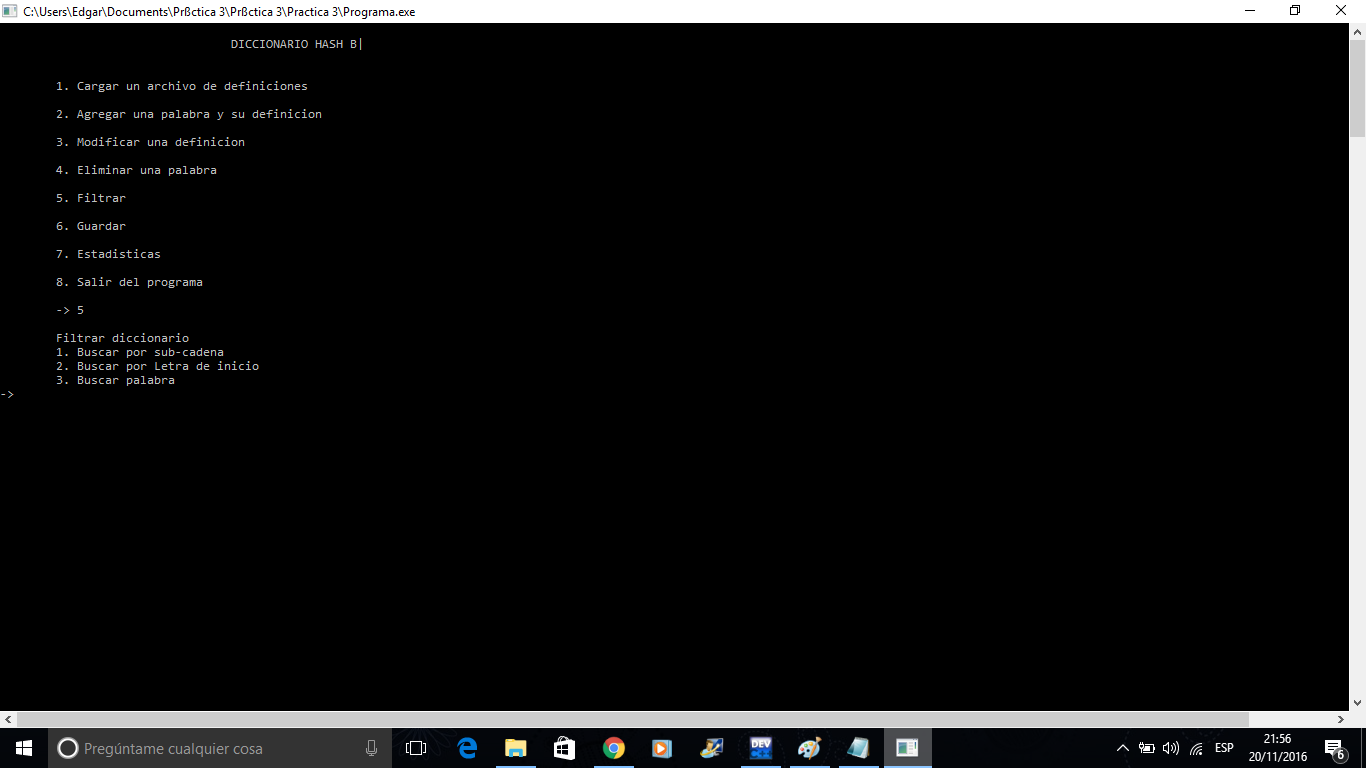
Aquí se ha buscado correctamente la palabra y se ha dado su definición.

 Figura 8

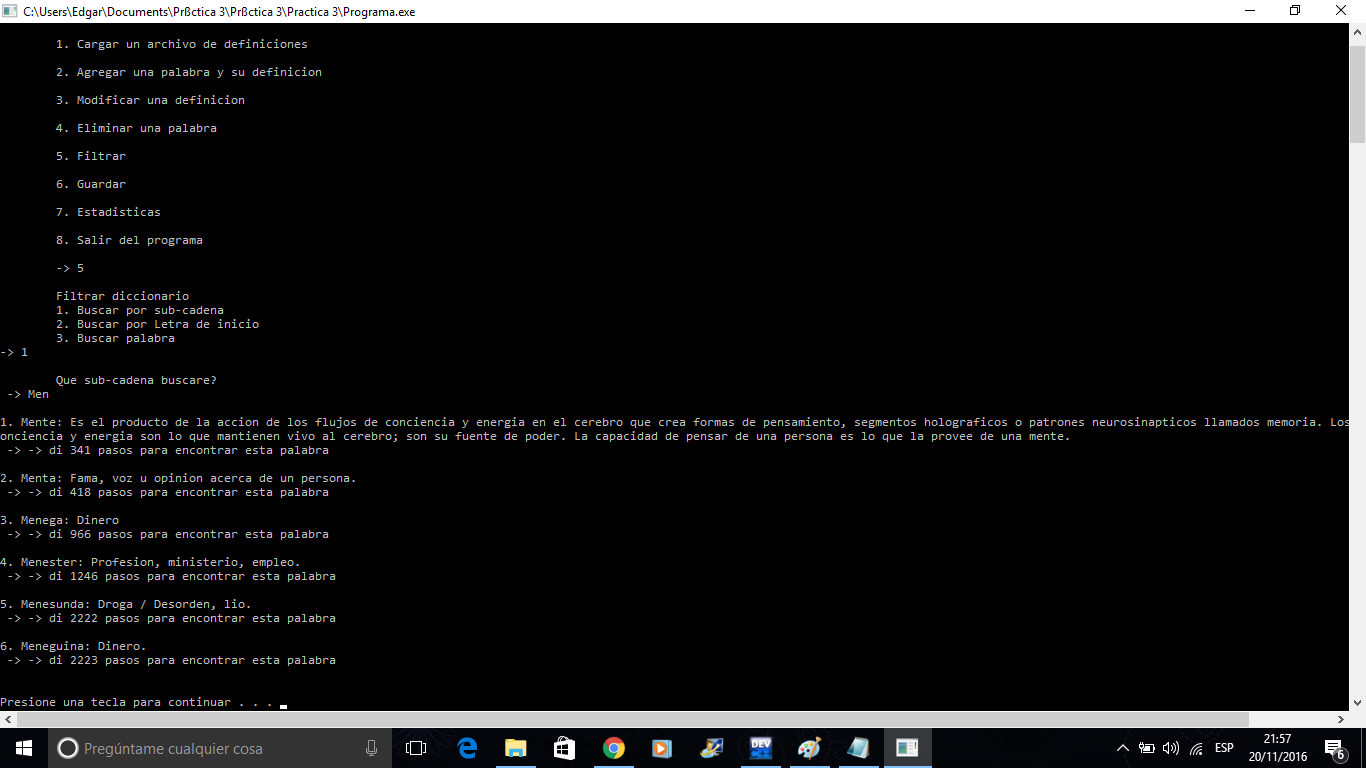
Ahora bien, aquí indica que esa palabra no se encuentra incluida en el diccionario.

 Figura 9

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

 Figura 10

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

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

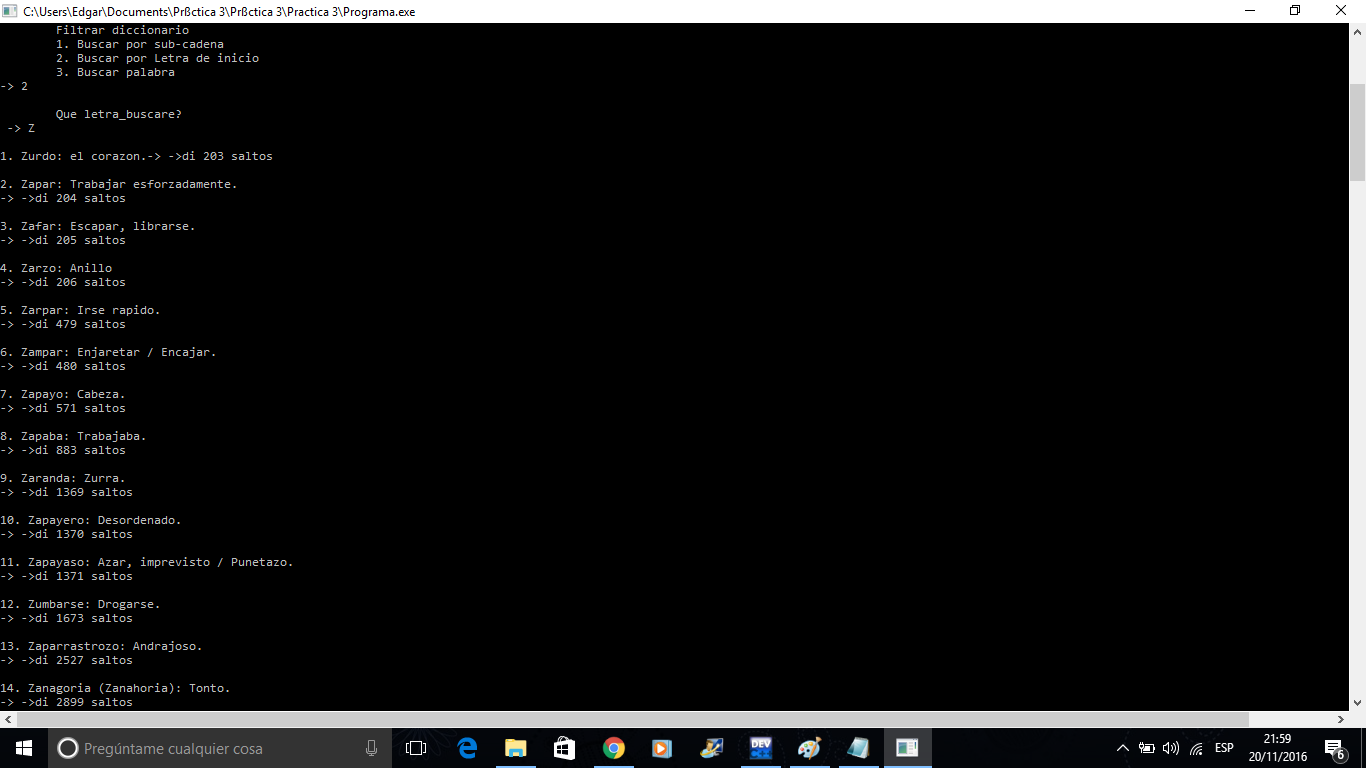
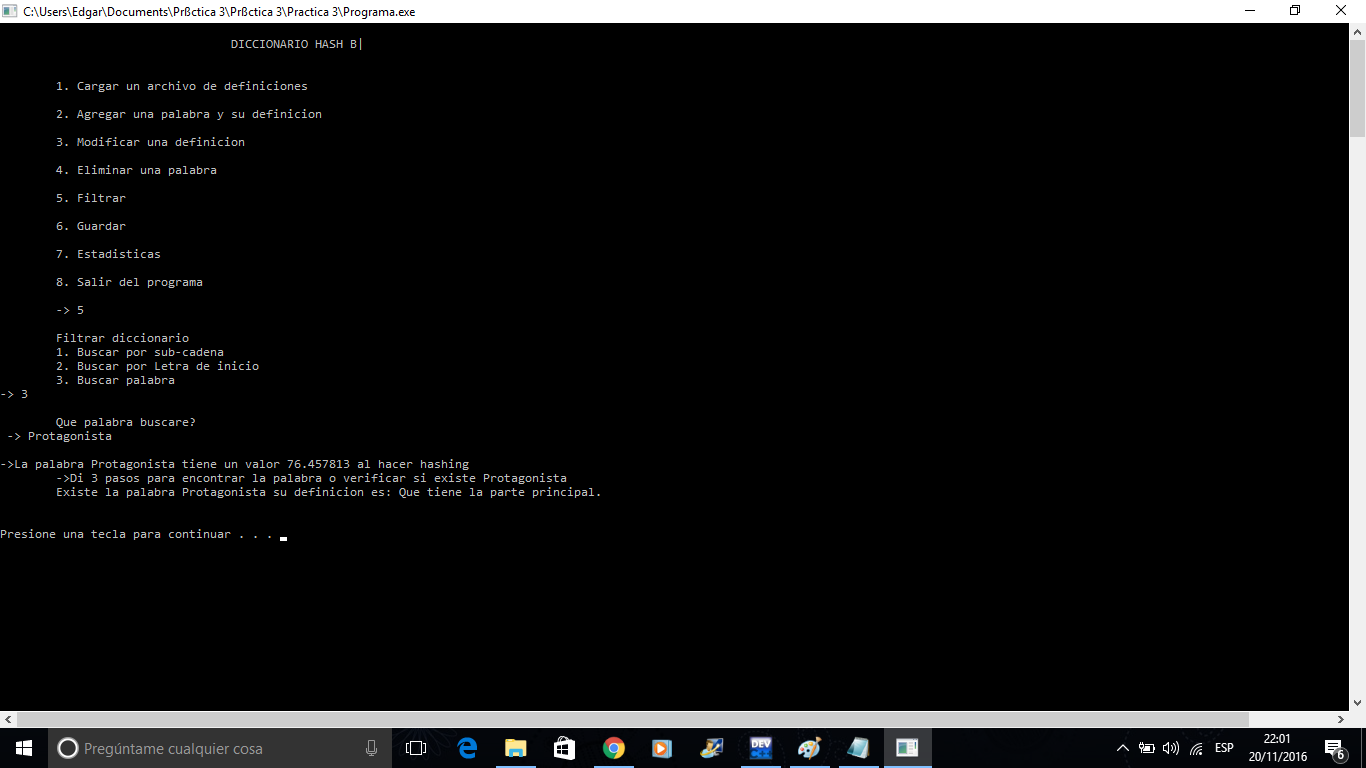
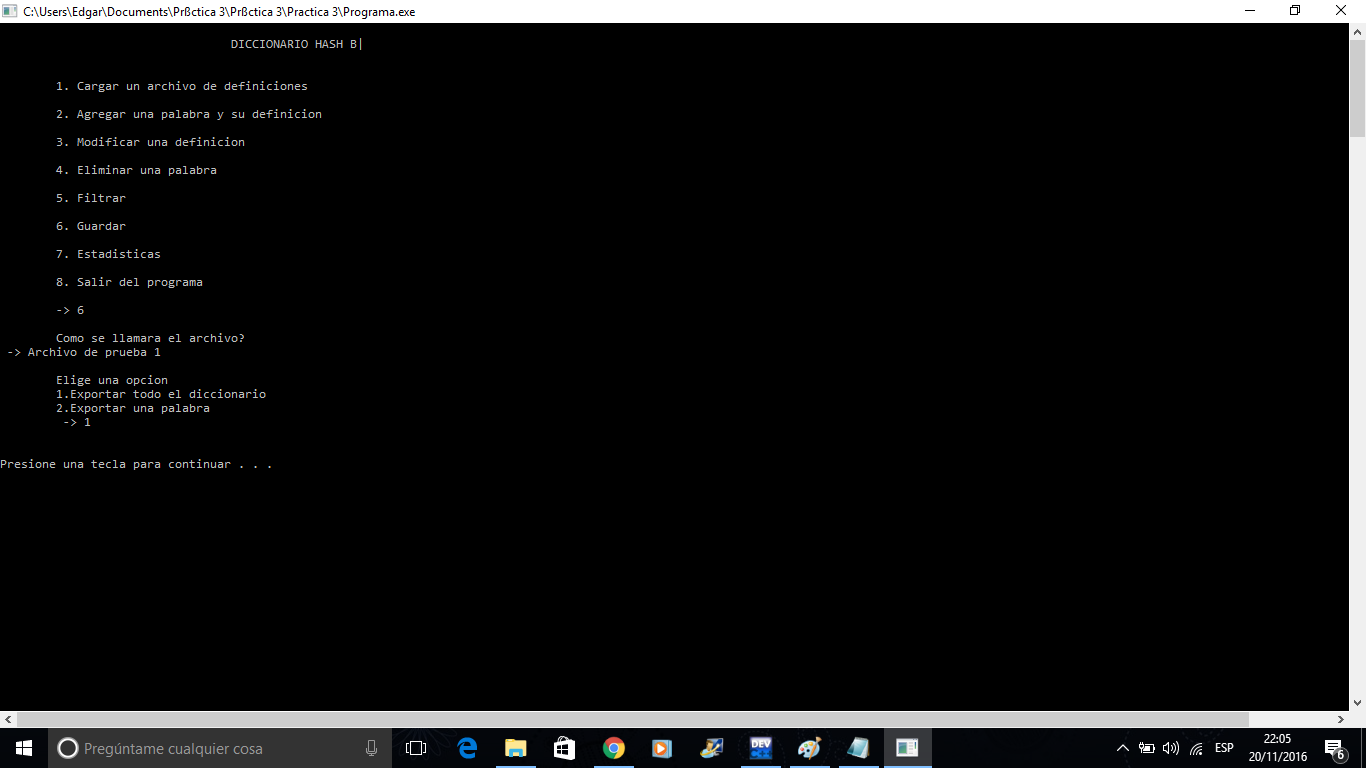


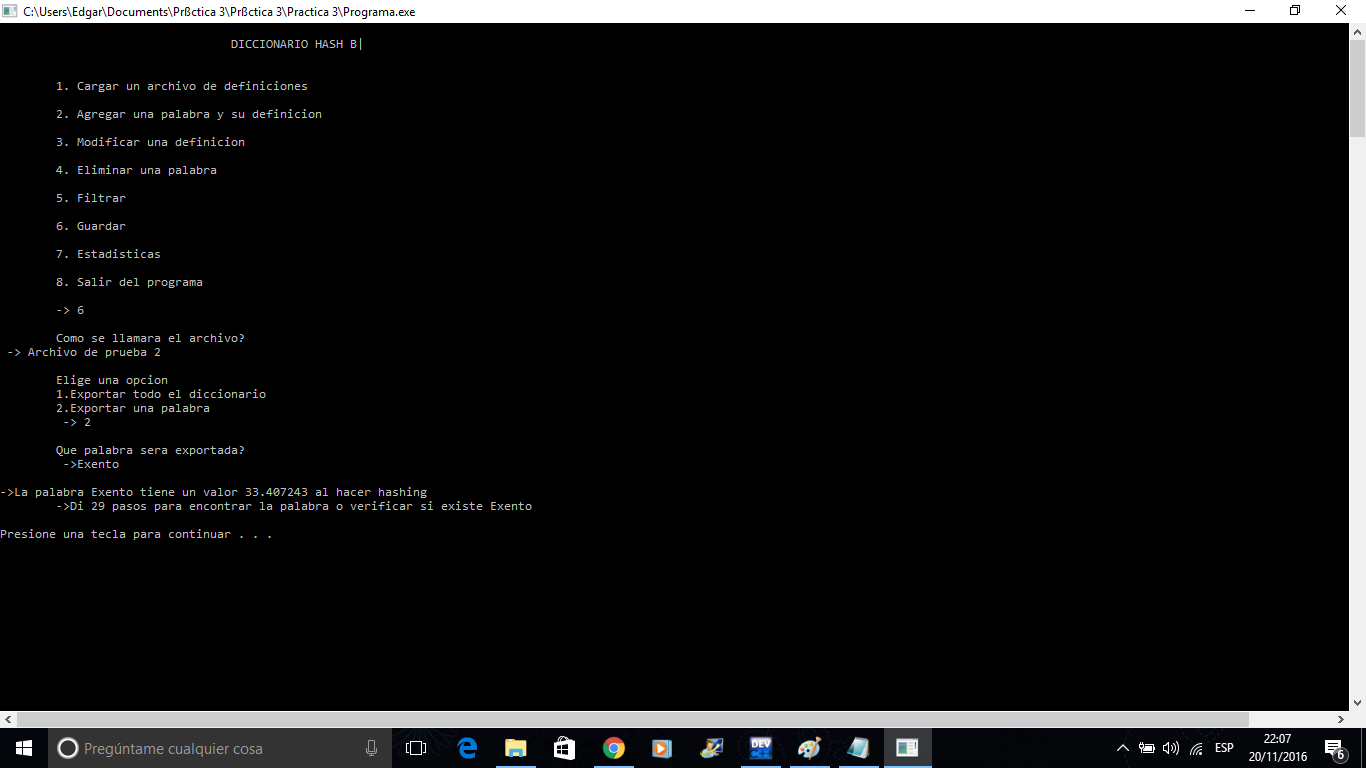
Figura 12. Al igual que la anterior la letra tiene diversos resultados.

 Figura 13

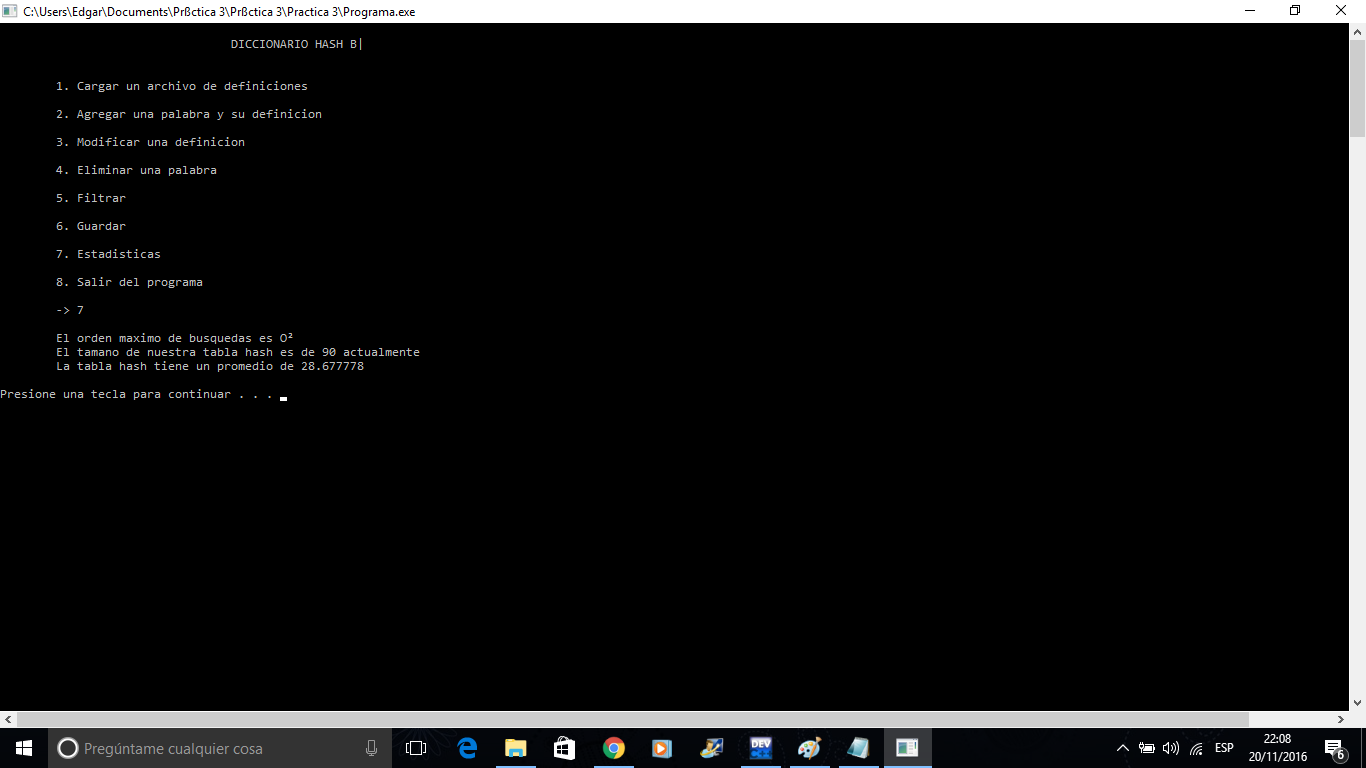
Al introducir la palabra, se encontró solo un resultado, debido a que solo existía una clave similar a la de ella.

 Figura 14

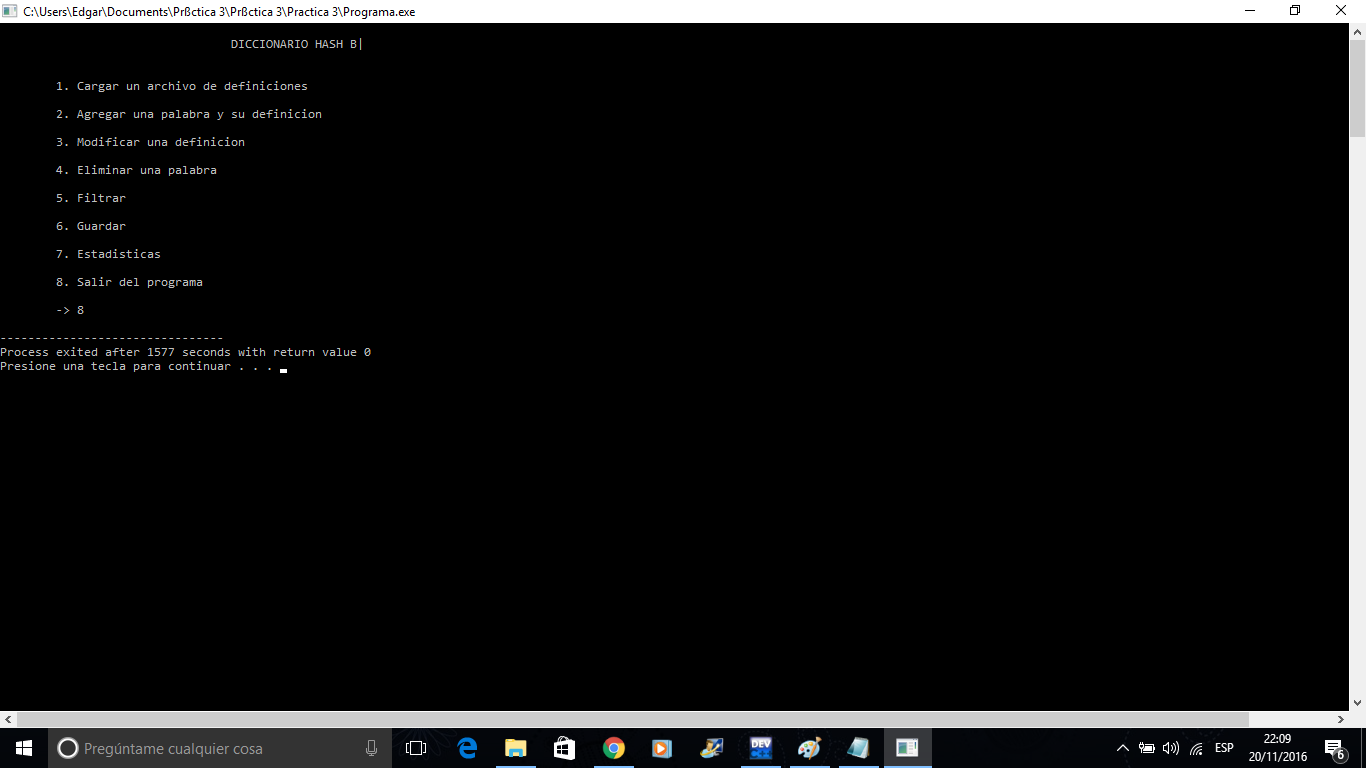
Aquí observamos que se guardara un archivo, y de nuevo, ofrecen dos opciones al usuario. Se elige la uno, por lo tanto e exporta el archivo que desee.

 Figura 15

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

 Figura 16

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

 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&nbsp; 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 encuenta 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 &quot;colisiones&quot; 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.

ANEXO

Fragmento del Programa para el Diccionario

// PROGRAMA PRINCIPAL

void main**(**void**)**

**{**

int i**;**// se declara i

lista tabla\_hash**[**1000**];**//se crea una tabla de 1000

**for(**i**=**0**;** i**<**1000**;** i**++)**// el for termina cuando ya no se cumple la condicion

Initialize**(&**tabla\_hash**[**i**]);**//se inicializa la tabla cerada

Leer\_Archivos**(**lunfardo**,** **&**tabla\_hash**);**// llama a la funcion

system**(**"CLS"**);**//limpia pantalla

//Estadisticas(&tabla\_hash);

Menu**(&**tabla\_hash**);**// //llama a la funcion

**}**

/\*

void Menu(lista\* tabla\_hash)

Descripcion:Da opciones al usuario de lo que desea hacer

Recibe: lista\* tabla\_hash

\*/

void Menu**(**lista**\*** tabla\_hash**)**

**{**

char ruta**[**250**]** **=** ""**;**

elemento mi\_elemento**;**//se declara mi\_elemento

strcpy**(**mi\_elemento**.**p**,** ""**);**

strcpy**(**mi\_elemento**.**d**,** ""**);**

printf**(**"\n\t\t\t\t DICCIONARIO HASH B|\n\n\n"**);**//se imprime a pantalla

int x **=** 0**;**

printf**(**"\t1. Cargar un archivo de definiciones \n\n"**);**//se imprime a pantalla

printf**(**"\t2. Agregar una palabra y su definicion \n\n"**);**//se imprime a pantalla

printf**(**"\t3. Modificar una definicion \n\n"**);**//se imprime a pantalla

printf**(**"\t4. Eliminar una palabra \n\n"**);**//se imprime a pantalla

printf**(**"\t5. Filtrar \n\n"**);**//se imprime a pantalla

printf**(**"\t6. Guardar \n\n"**);**//se imprime a pantalla

printf**(**"\t7. Estadisticas \n\n"**);**//se imprime a pantalla

printf**(**"\t8. Salir del programa \n\n\t-> "**);**//se imprime a pantalla

scanf**(**"%d"**,** **&**x**);**

**switch** **(**x**)**

**{**

**case** 1**:** //printf("Lala 1 \n\n");

fflush**(**stdin**);**//elimina la basura

printf**(**"\nCargare un archivo de definiciones escribe el nombre del archivo \n->"**);**//se imprime a pantalla

gets**(**ruta**);**//almacena

strcat**(**ruta**,** ".txt"**);**

Leer\_Archivos**(**ruta**,** tabla\_hash**);**

printf**(**"\n\n"**);**//se imprime a pantalla

system**(**"PAUSE"**);**//termina el proceso

system**(**"CLS"**);**//limpia pantalla

Menu**(**tabla\_hash**);**//llama la funcion

**break;**

**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 proporcionadapor 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\tEsa 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 caracter**;**

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\_buscare? \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 buscare? \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 palbra, 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 O%c"**,** 253**);**//se imprime a pantalla

printf**(**"\n\tEl tamano 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