**Instituto Politécnico Nacional**

**Escuela Superior de Cómputo**

Alumno: Vargas Romero Erick Efraín

Profesor: Franco Martínez Edgardo Adrián

Unidad de aprendizaje: Estructuras de datos

Tarea 04: Programa con archivos (Gestión de empleados)

Secuencia: 1CM9

Semestre: 2016 – 2017 “A”

**Índice**

Problema 3

Pruebas 3

Prueba 1 3

Prueba 2 6

Prueba 3 7

Prueba 4 8

Prueba 5 9

Prueba 6 10

Prueba 7 11

Prueba 8 12

Prueba 9 14

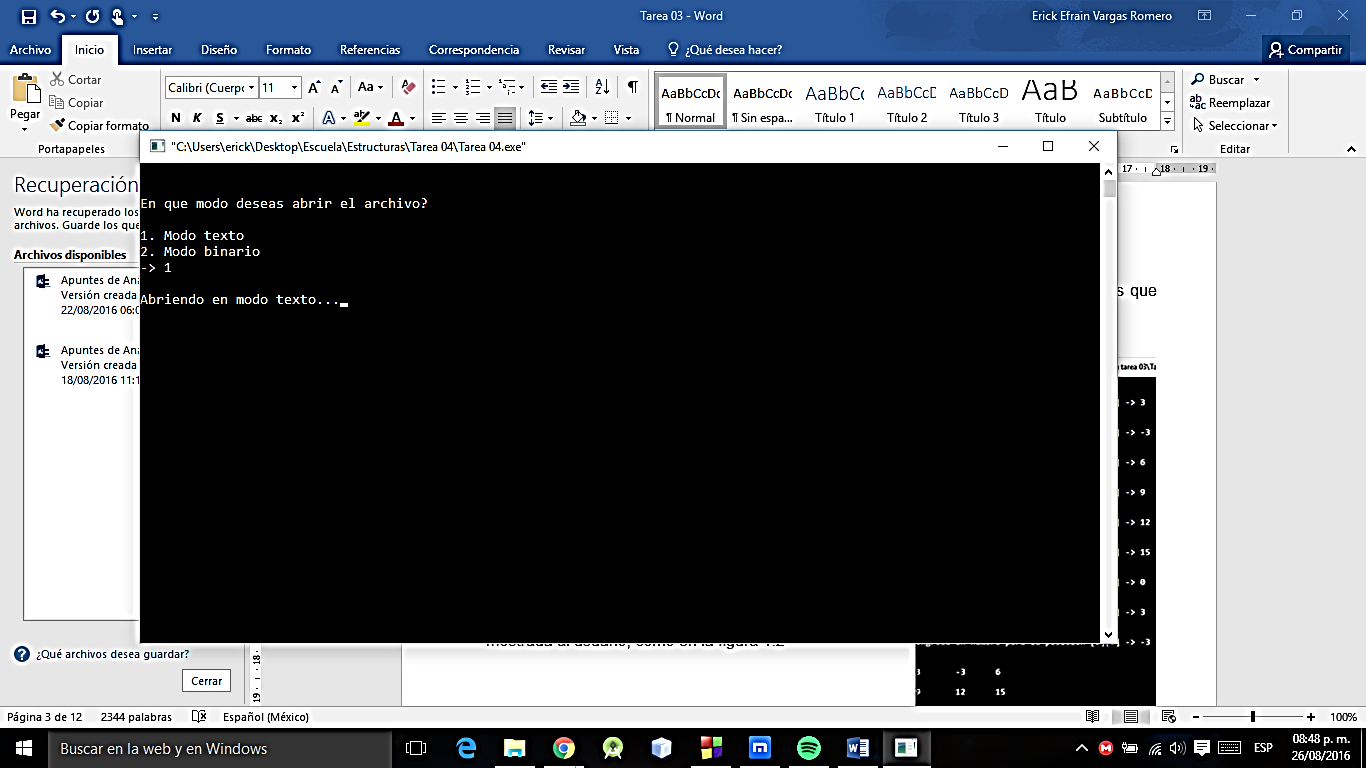
Implementación 15

# **Problema**

Modifica el programa del ejemplo para que el programa pueda realizar, altas, bajas y cambios de los empleados, además de los respaldos en modo texto y binario según el usuario lo solicite. Crear un menú para operar con el programa. El programa deberá de ser capaz de salir y almacenar la lista de empleados para que al iniciar nuevamente mantenga la información de ellos.

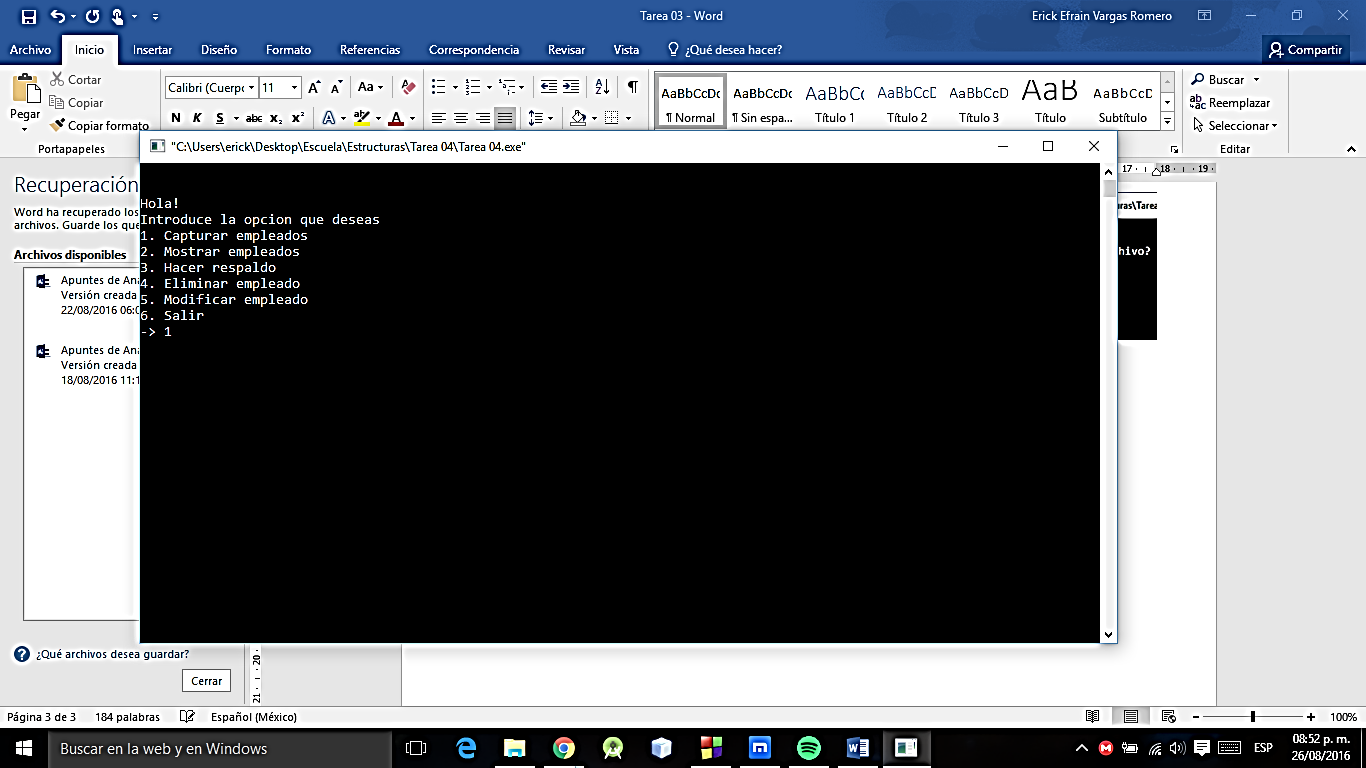
# **Pruebas**

## **Prueba 1**



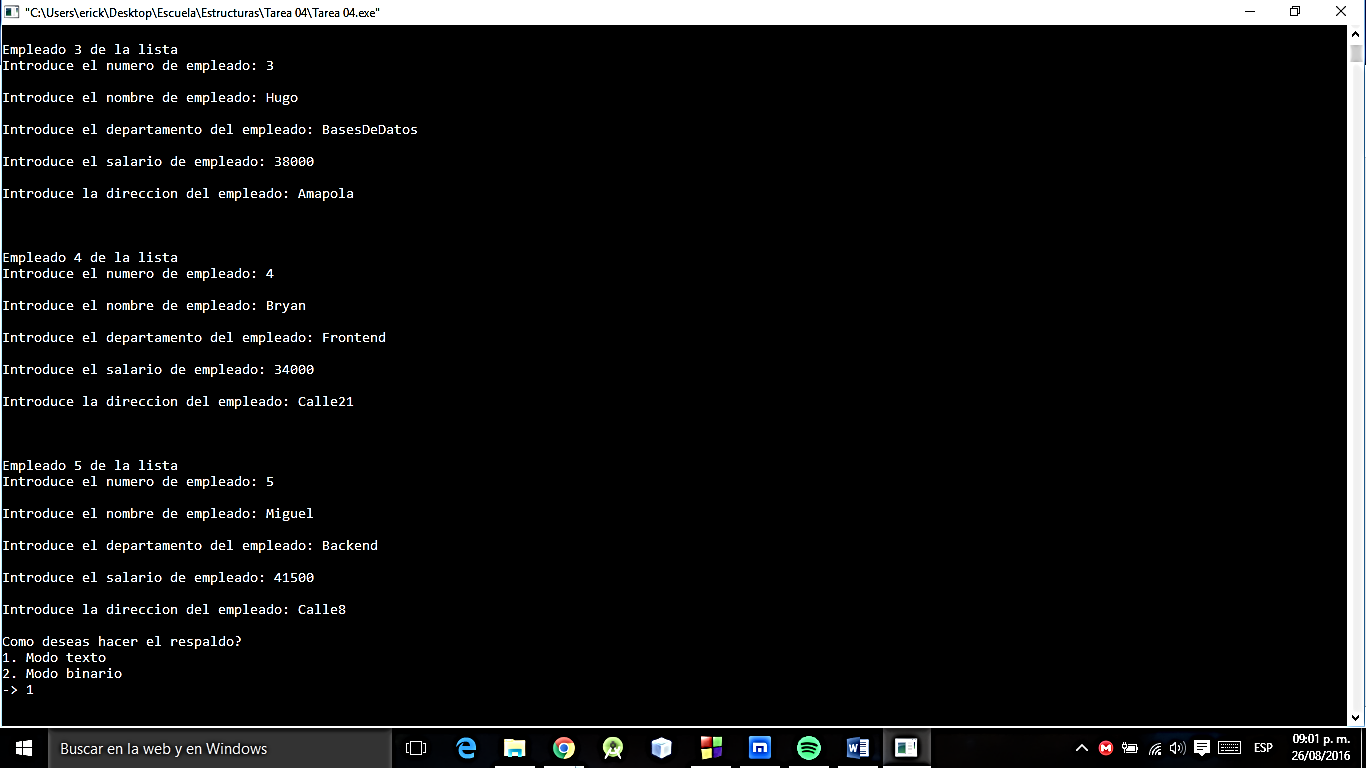
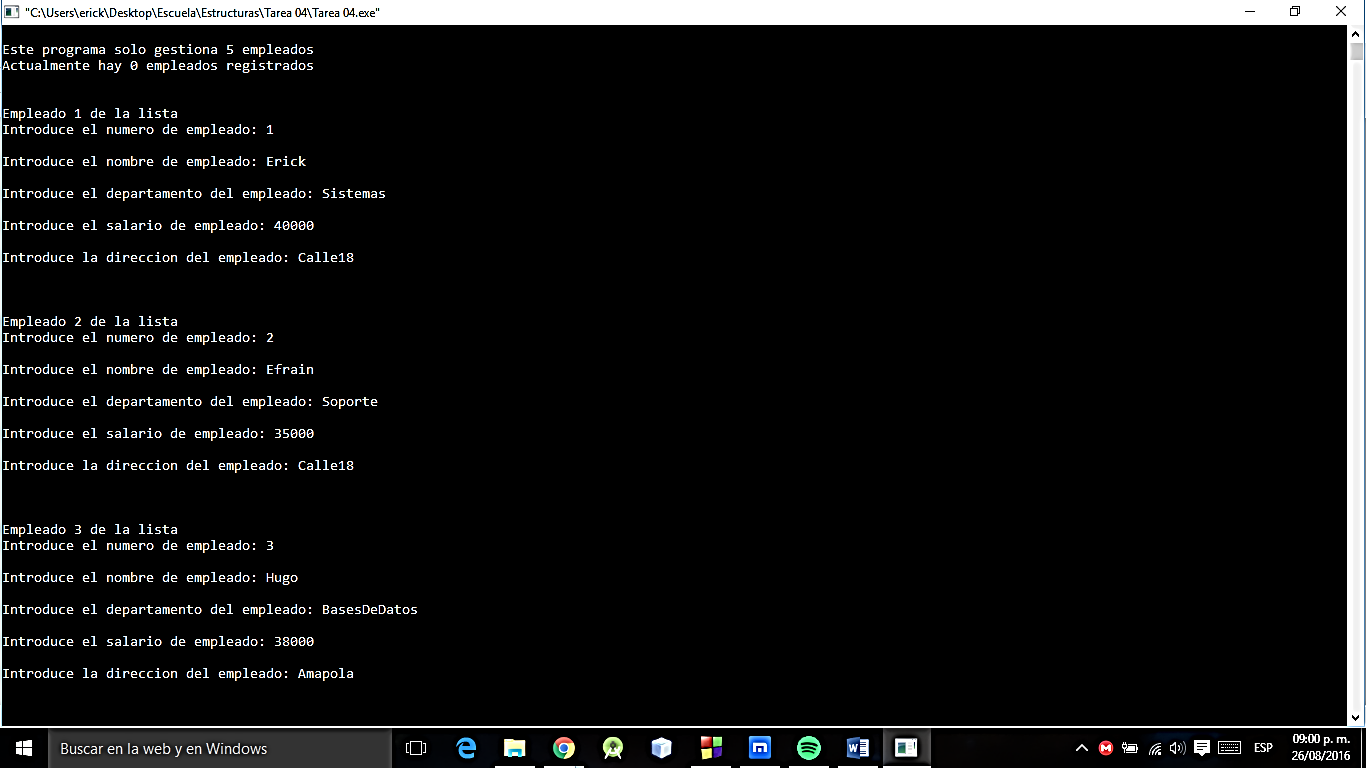
En la primera prueba como se puede observar en la imagen 1.1, al iniciar el programa (En todos los casos) lo primero que debemos realizar en el programa es qué modo deseamos abrir el archivo, en modo texto o binario, en este caso lo hacemos en modo binario (Se ejecuta la función OpenHow, y posteriormente se hace la selección del modo)

**Figura 1.1**



Una vez hecho lo anterior, el programa nos envía al main(), en este lo que hacemos es elegir que realizaremos primero, en este caso no tenemos aún empleados registrados, no hay problema si entramos a las demás opciones ya que hacemos una evaluación en todos los casos para saber si hay o no datos almacenados en el arreglo empleado. En esta ocasión lo que seleccionaremos es la opción uno, para almacenar usuarios (**Nota:** Para esta prueba la constante NUM\_EMP la definimos con un valor de 5) Figura 1.2

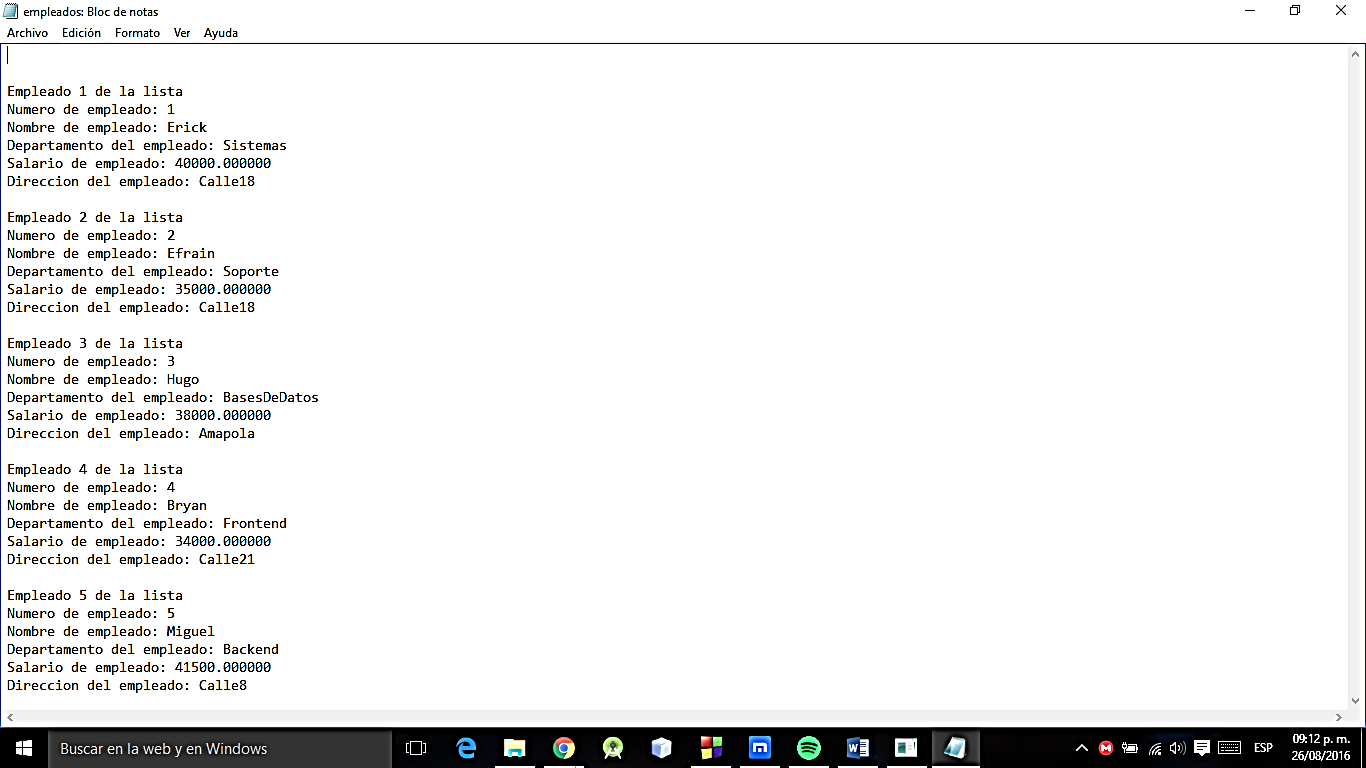
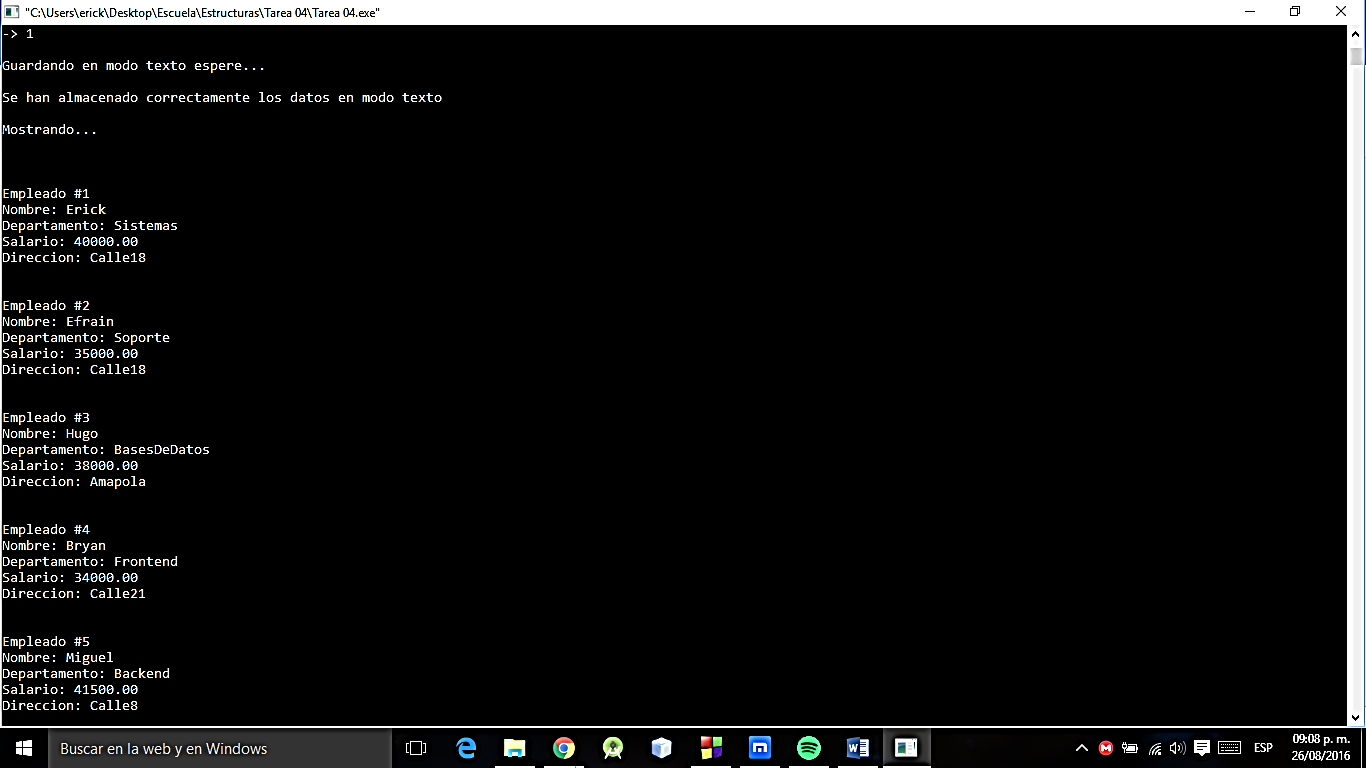
**Figura 1.2**



**Figura 1.4**

**Figura 1.3**

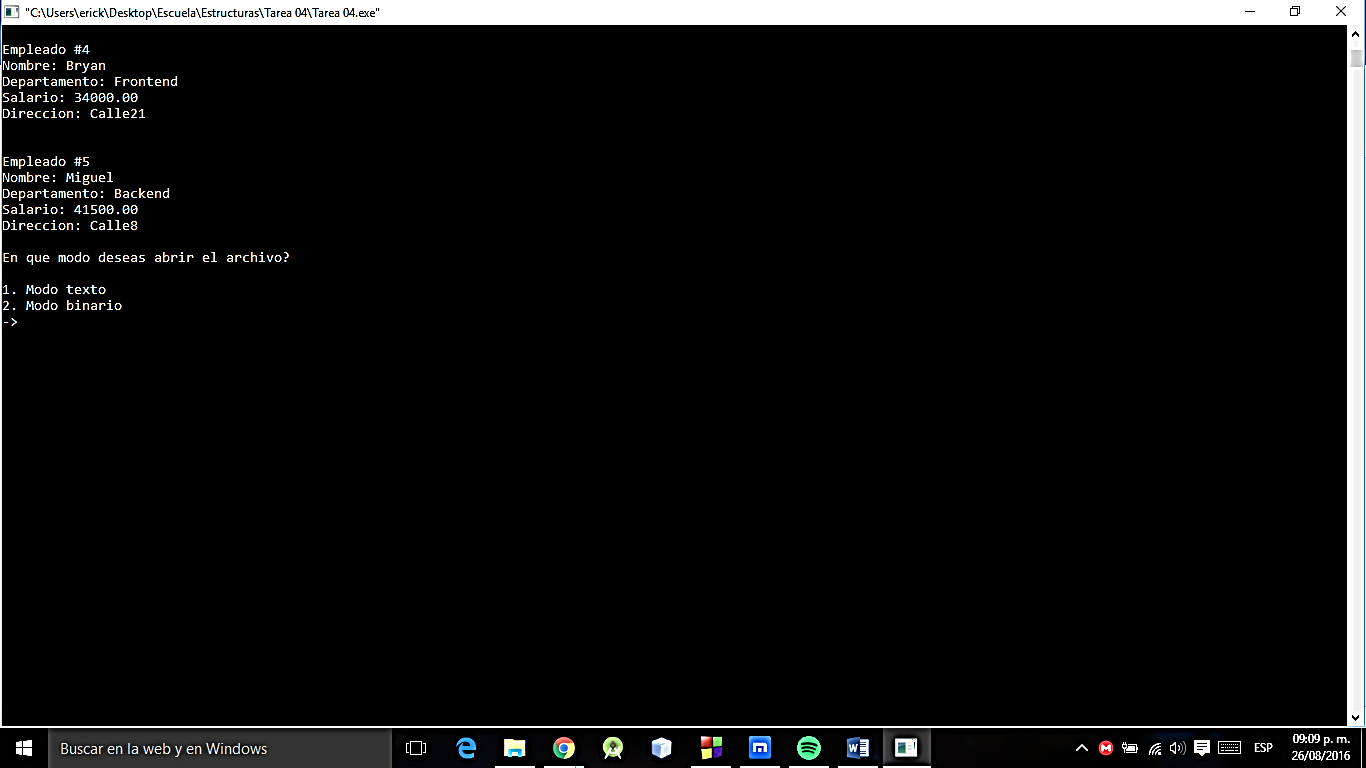
Como podemos observar, ingresamos los datos de nuestros empleados, ingresando primeramente el número del empleado, el nombre del empleado, el departamento al que pertenece, el salario que gana, y la dirección en la que vive, (**Nota:** La función scanf no lee espacios, por eso en palabras en las que debería se escribió con notación camello). Una vez que hemos ingresado los datos de los 5 empleados en este caso, se ejecuta una función llamada CreateBackUp, hace un respaldo de lo que ya está almacenado en nuestra estructura y dándonos a elegir como deseamos guardar el archivo si en modo texto o modo binario. (Figuras 1.3, y 1.4)



**Figura 1.6**

**Figura 1.5**

En esta sección, una vez que se han almacenado los datos, se ejecuta el método LeeEmpleadosTexto, en este caso es así porque almacenamos los datos en modo texto, si hubiese sido lo contrario los datos serían leídos en modo binario. Como observamos, efectivamente, se han almacenado los datos en son mostrados (Figura 1.5) tal cual los hemos almacenado, para corroborar, hago el muestreo del contenido del archivo empleados.txt (Figura 1.6) Y de igual manera cómo podemos observar se han almacenado los archivos como lo hicimos en el programa.



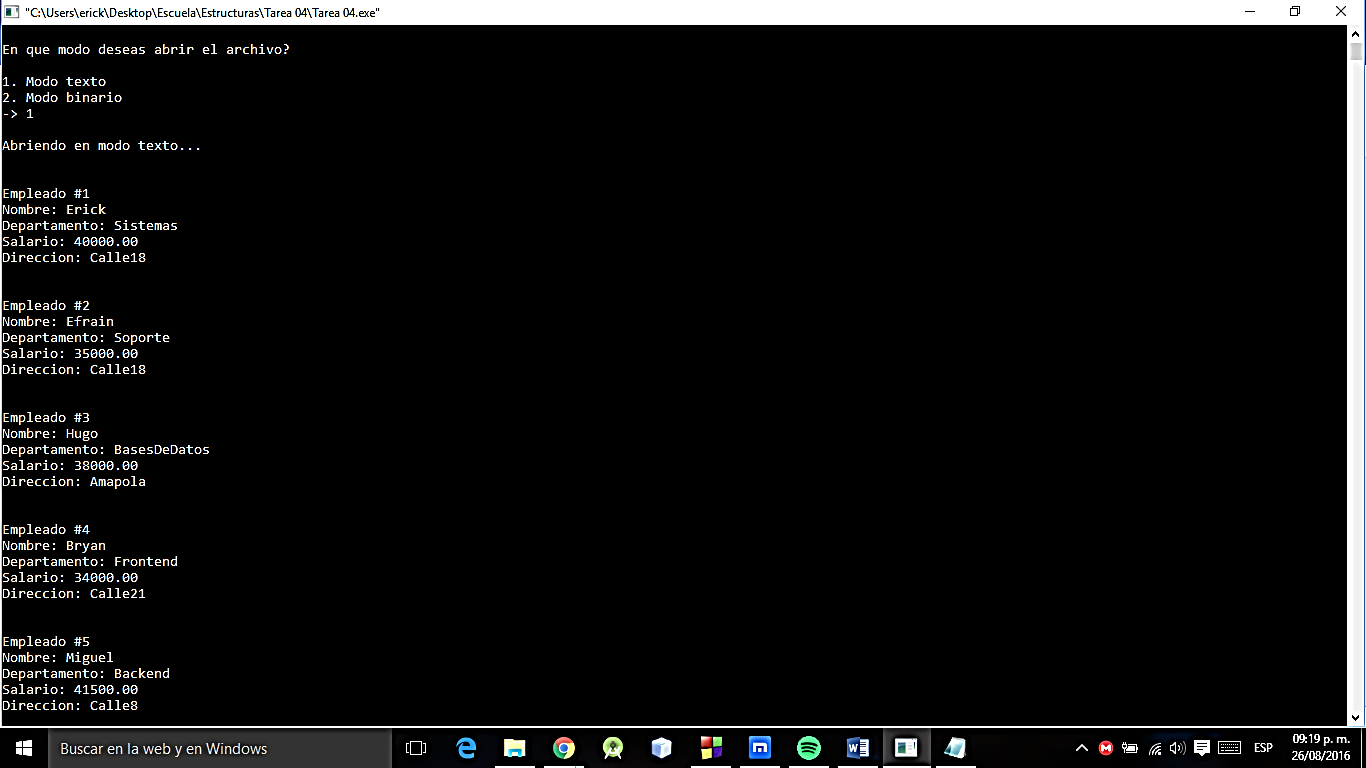
Como podemos observar en la figura 1.7 al haber ocurrido lo anterior, invocamos nuevamente al main, para poder seguir realizando operaciones haciendo selección en el pequeño menú que hemos construido como se mostró en la figura1.2

**Figura 1.7**

## **Prueba 2**

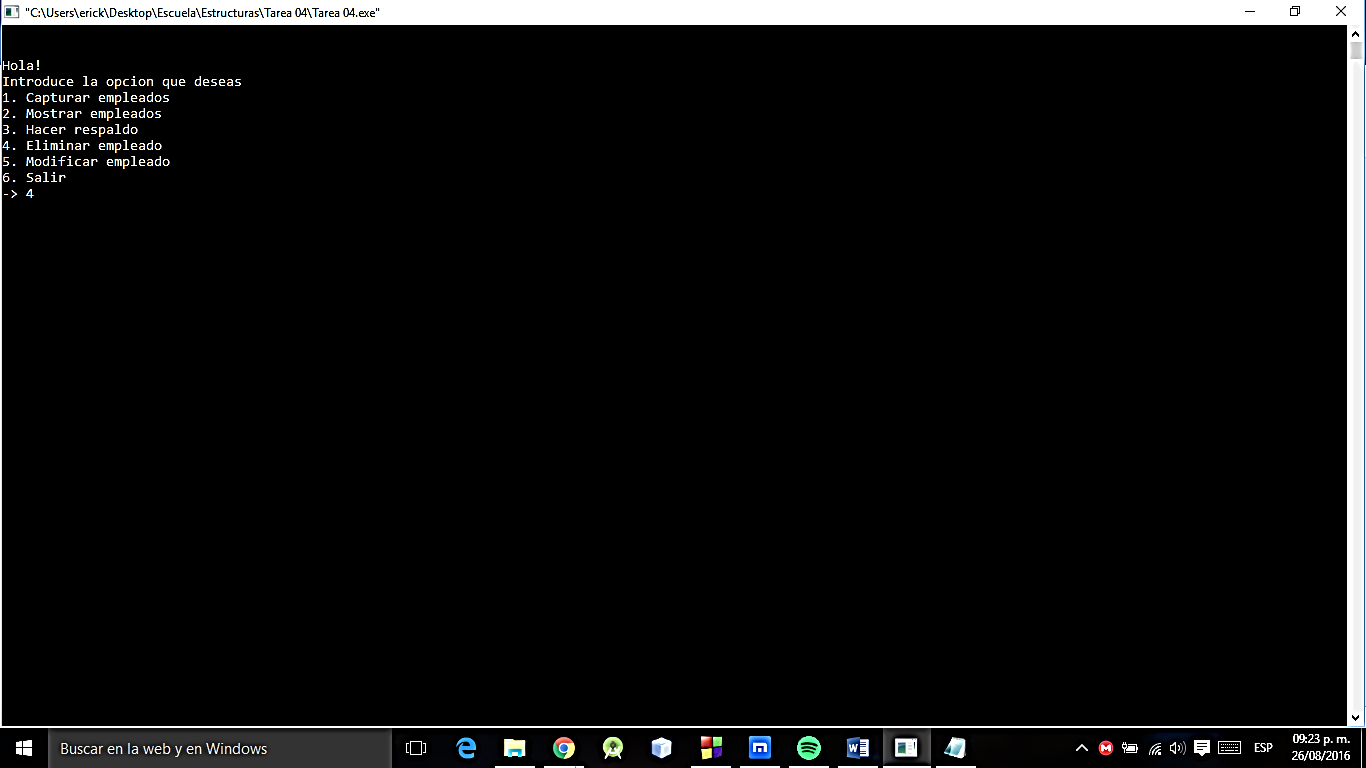
Para esta prueba mostraremos algo sencillo y es como el programa puede mostrar desde el menú lo que está almacenado, en este caso hacemos selección de la opción 2 de nuestro menú figura 2.1

**Figura 2.1**



Como podemos observar en la figura 2.2, al igual que en la figura 1.5, seleccionamos el modo que deseamos mostrar, nuevamente elegimos el modo texto, y se hace el muestreo de lo que está almacenado en nuestro fichero empleados.txt, almacenado en nuestro arreglo empleado y finalmente mostrado al usuario

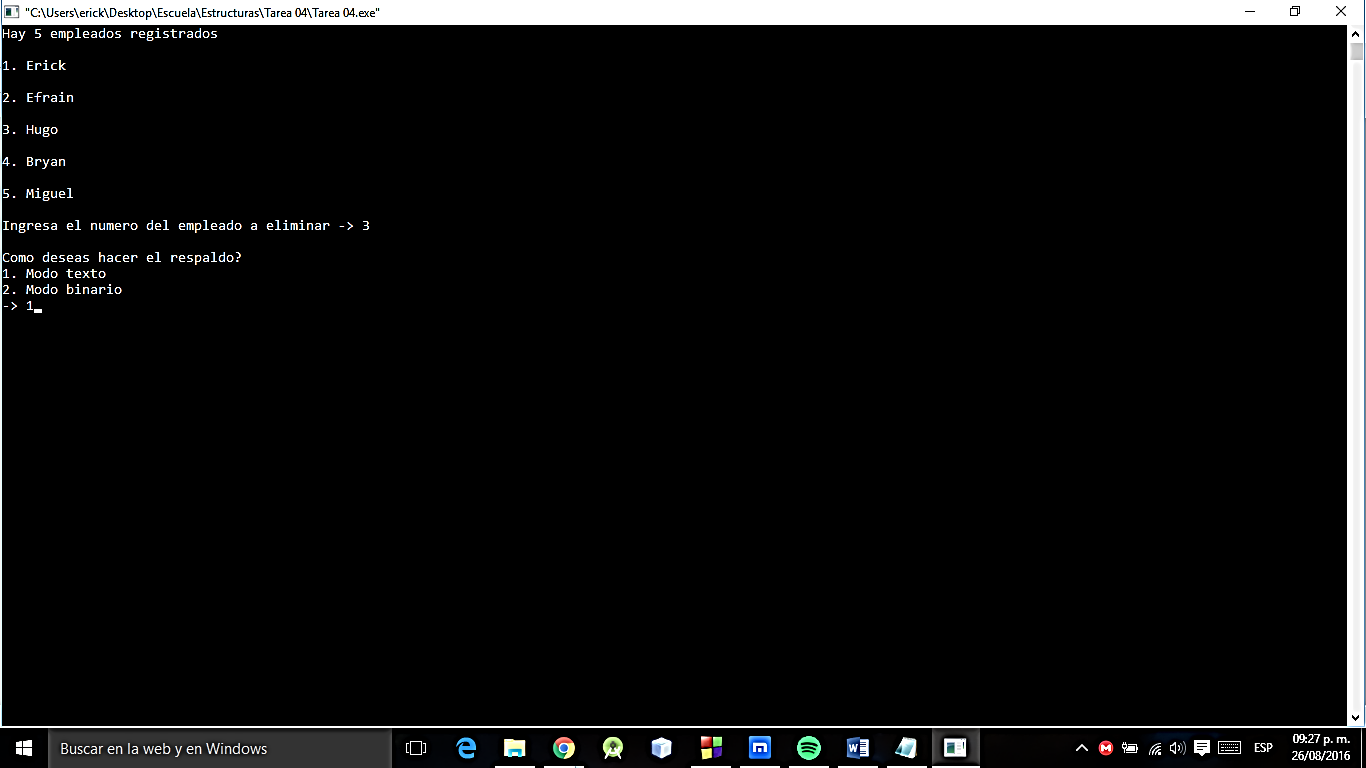
**Figura 2.2**



## **Prueba 3**

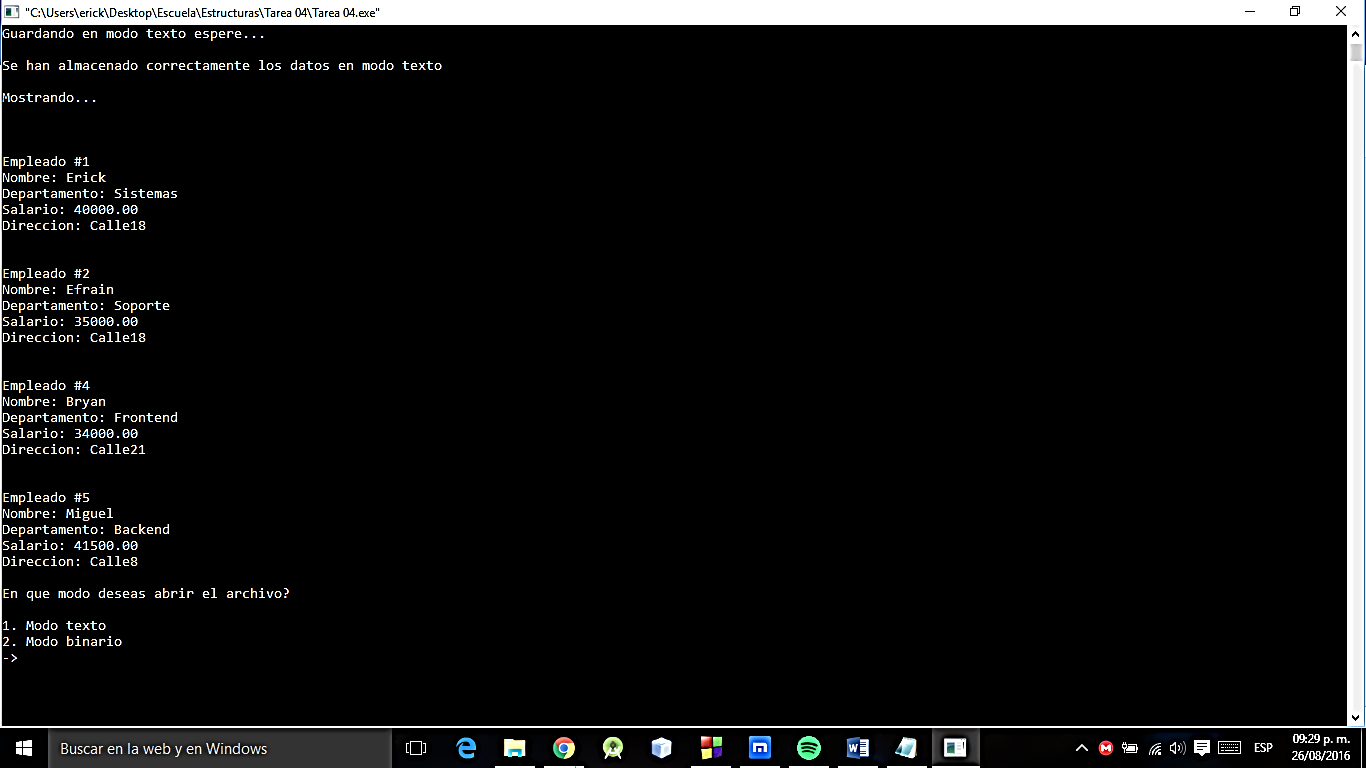
Nuevamente mostraré el pequeño menú que se ejecuta en el main del programa, y por ahora nos saltaremos la opción 3 que mostraremos más adelante para realizar operaciones con el modo binario, por ahora seleccionamos la opción 4 para eliminar un empleado (Figura 3.1)

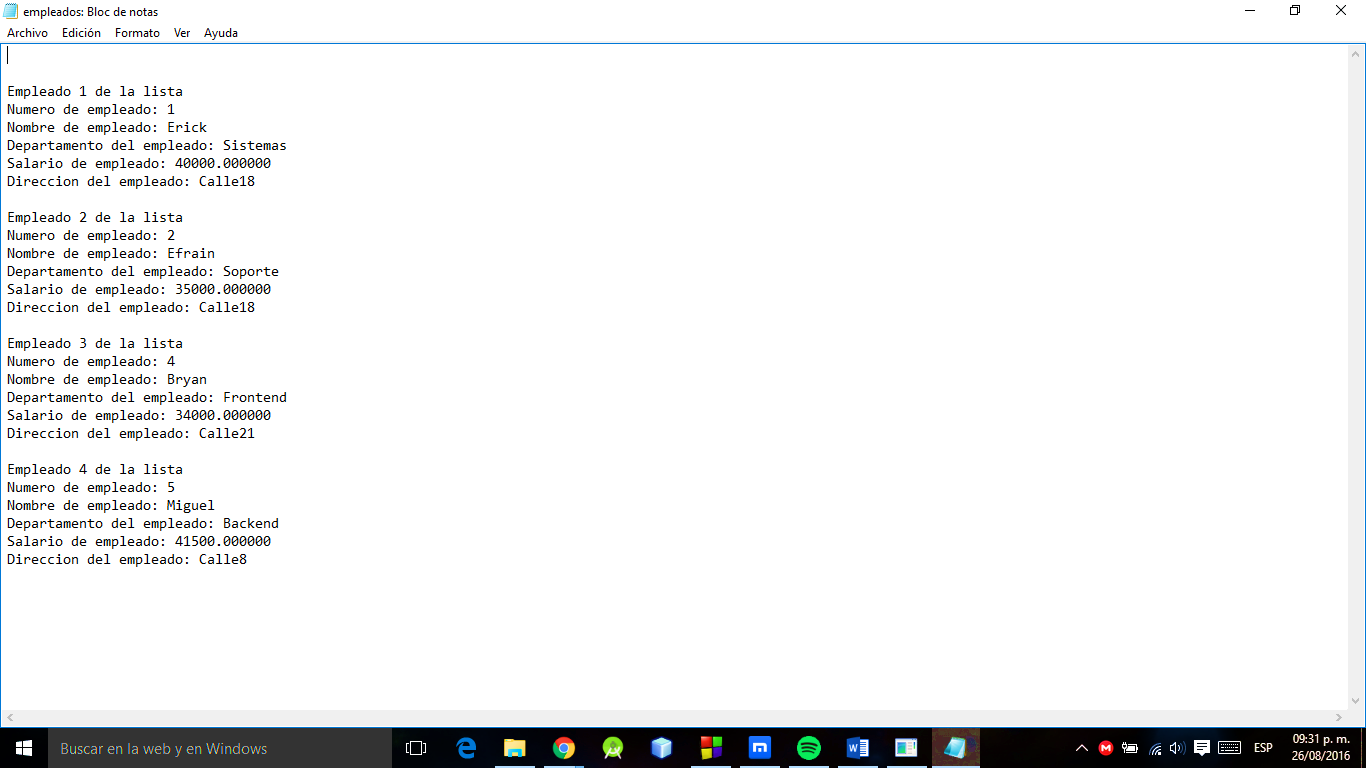
**Figura 3.1**



Como se puede apreciar en la figura 3.2 el programa primeramente muestra el número de usuarios registrados, esto lo hace usando una función llamada isEmpty que nos retorna el valor 0 o 1 dependiendo si hay algo almacenado o no. Lo que se muestra después es el nombre de los empleados y finalmente el usuario ingresa cual desea eliminar, en este caso eliminamos al empleado 3, y hacemos el respaldo, en este caso, nuevamente en modo texto.

**Figura 3.2**





Como se puede observar en la figura 3.3 se ha eliminado el usuario llamado Hugo, y ya ha sido almacenado en nuestro fichero, para corroborar, anexaré una captura (Figura 3.4) y en efecto, ha sido eliminado

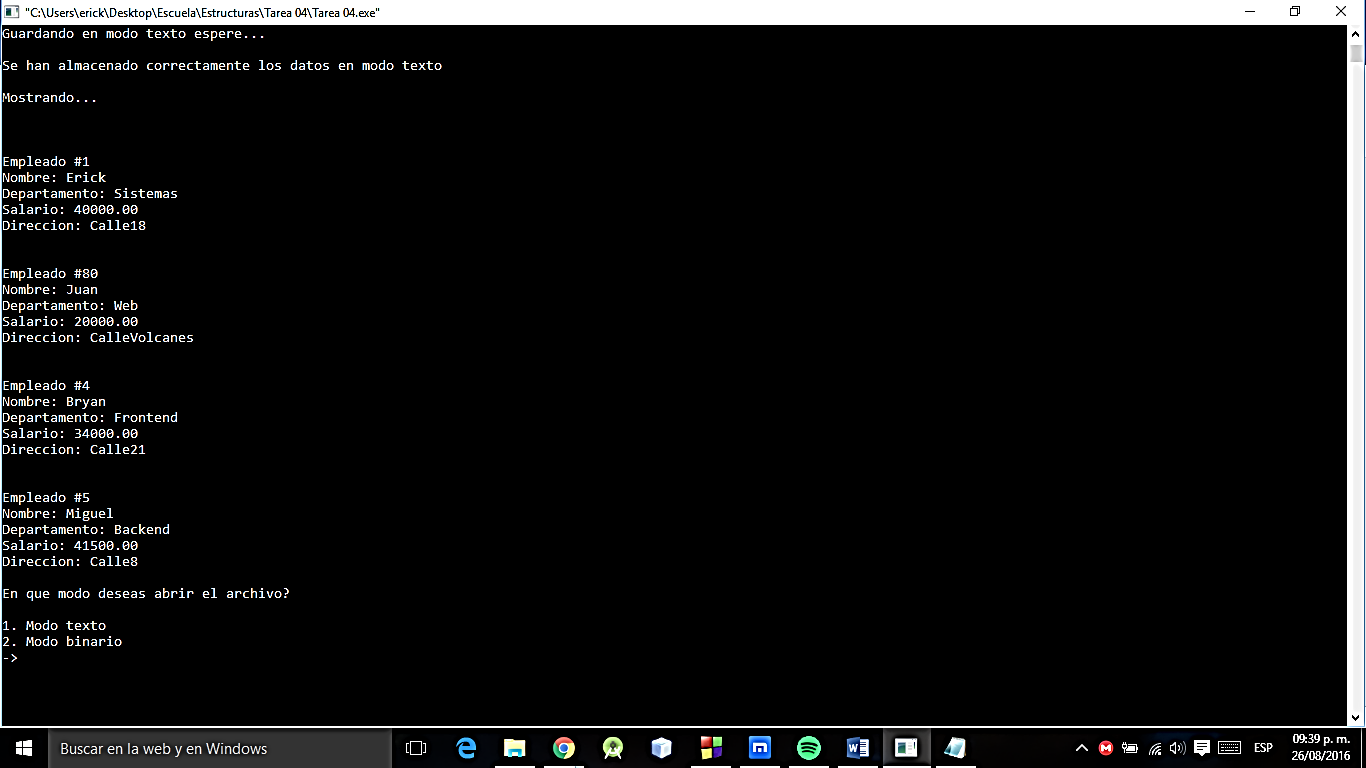
**Figura 3.4**

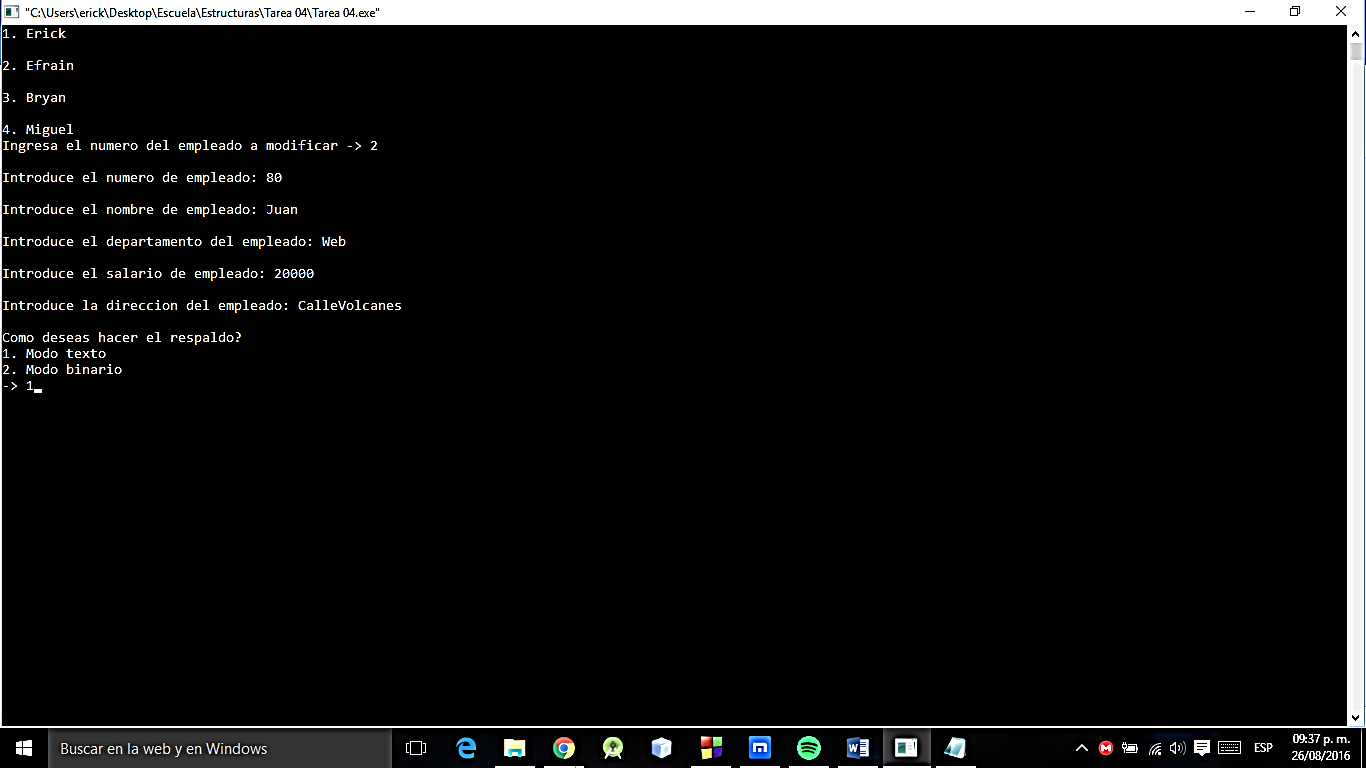
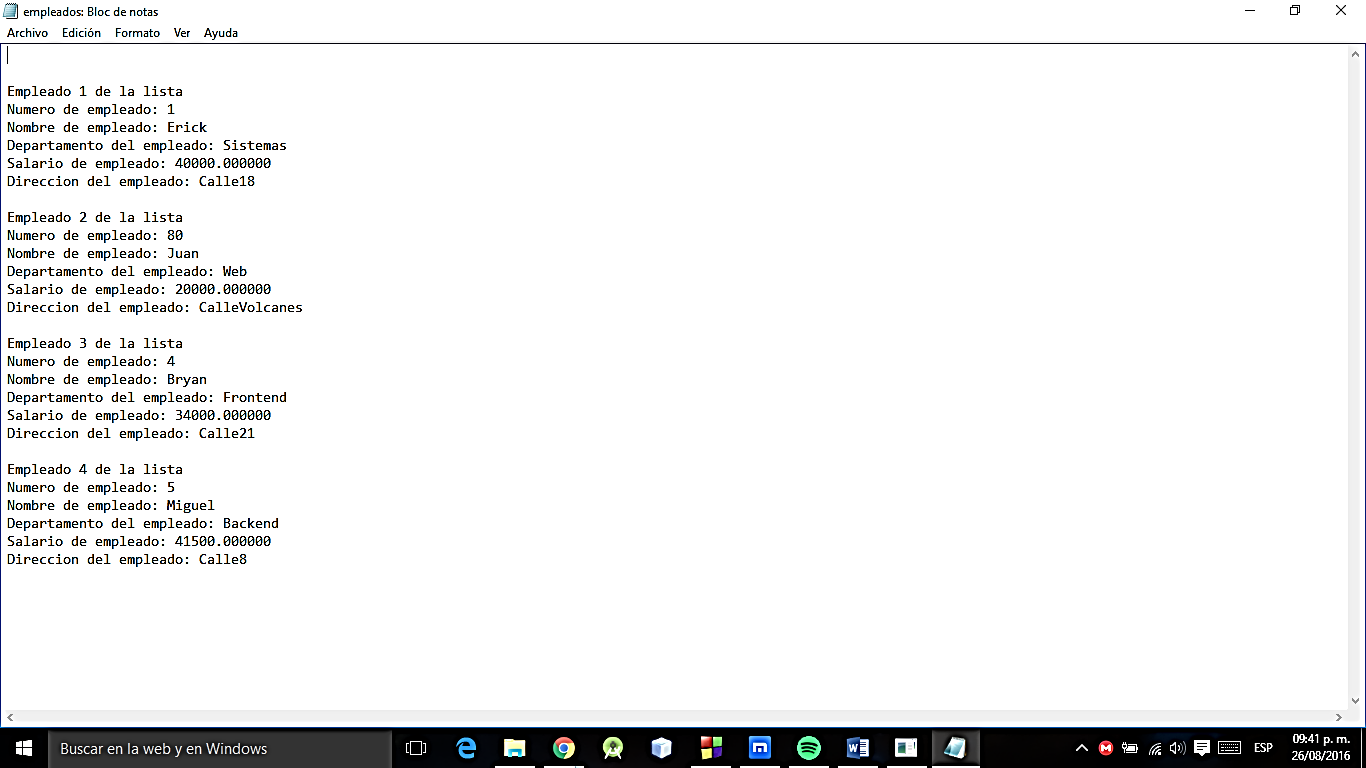
**Figura 3.3**

## **Prueba 4**

En esta ocasión en el menú ingresaremos la opción 5, para realizar la modificación de un empleado, recordemos que ya hemos eliminado a uno, así que debería el programa solo mostrarnos 4 empleados y no 5 como fue inicialmente. Figura 4.1

**Figura 4.1**





**Figura 4.4**

**Figura 4.3**

**Figura 4.2**

En la figura 4.2 podemos ver que al dar enter, lo primero que es mostrado es el nombre de los usuarios, para poder realizar la modificación, en este caso modificaremos al usuario 2, y le cambiaremos el número de empleado a 80, el nombre a Juan, el departamento a Web, el sueldo a 20000 y la dirección a Calle volcanes, finalmente hacemos el respaldo de esa información, que podemos apreciar en la figura 4.3, para corroborar, anexaré una imagen (Figura 4.4) que muestra cómo han sido modificados los datos, y cómo es posible comparar, son exactamente los mismos.

## **Prueba 5**

Ya hemos terminado de operar con el modo texto, pero nos resta el modo binario, primeramente, lo que vamos a hacer es hacer un respaldo de lo que ya hemos hecho anteriormente, tal como estaba, y “jugaremos” con esos datos (Figura 5.1)

**Figura 5.1**



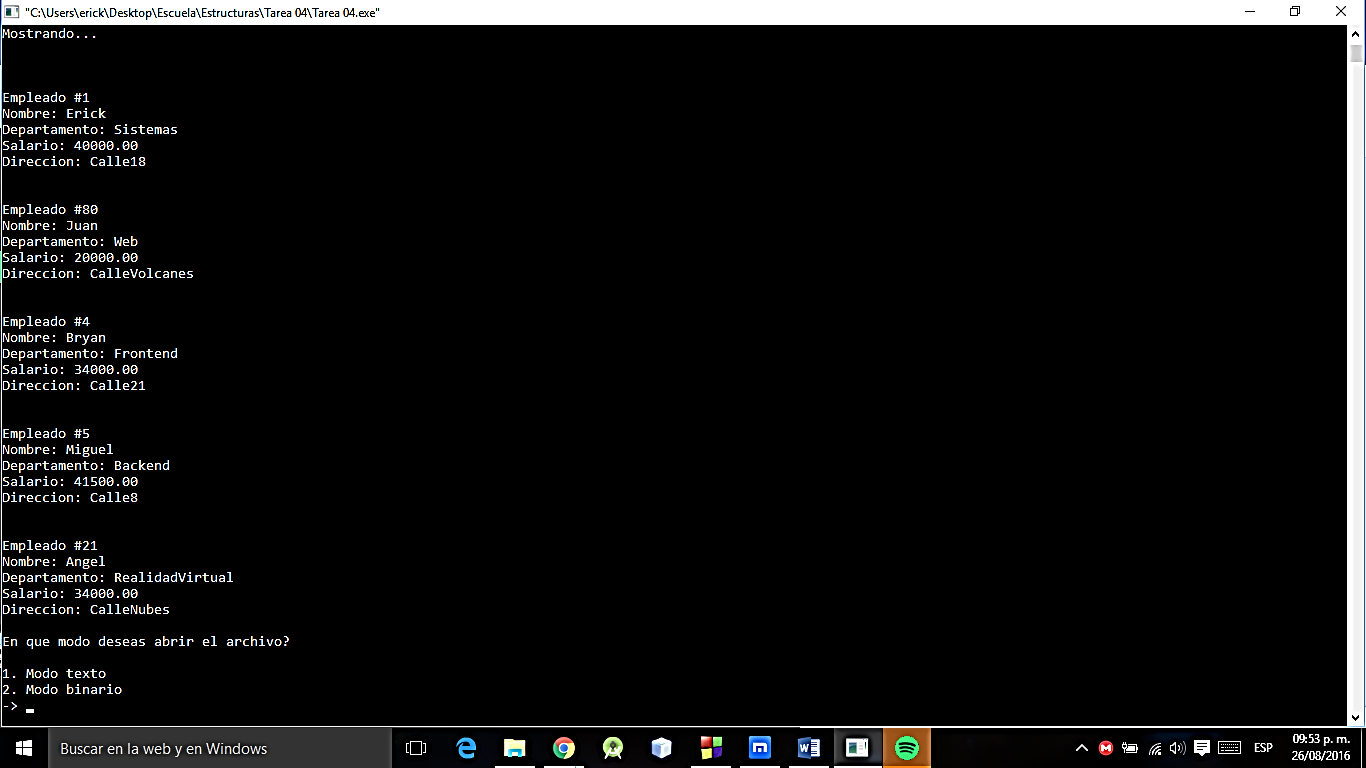
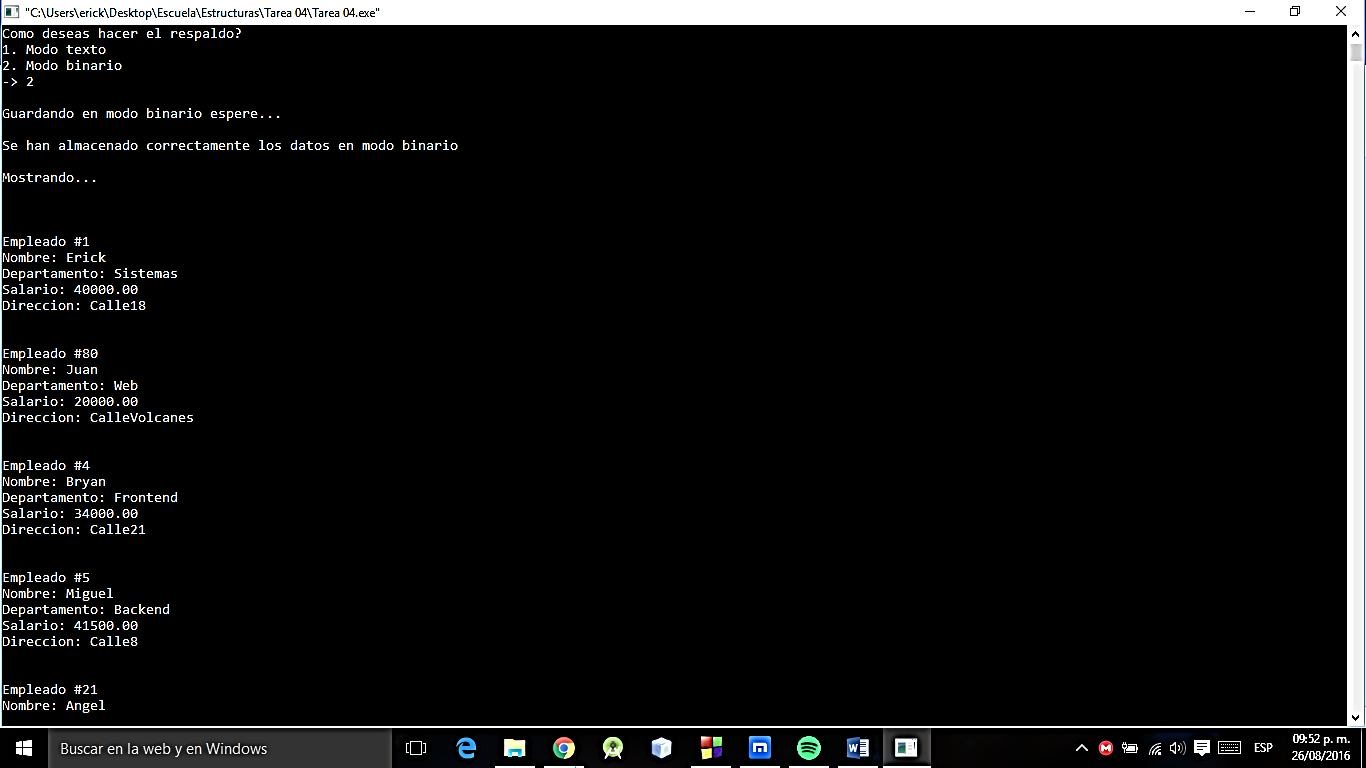
Como se observa en la figura 5.2, se ha creado el respaldo en modo binario de todo lo que teníamos anteriormente, incluyendo obviamente la modificación, ahora con estos 4 empleados registrados realizaremos el resto de operaciones.

**Figura 5.2**

## **Prueba 6**

Como recordaremos hemos eliminado 1 de los 5 empleados que el programa por esta ocasión gestiona, así que añadiremos en que nos resta, para eso seleccionaremos nuevamente la opción capturar empleados. Como podemos observar, el programa nos envía el mensaje que hay 4 empleados registrados de 5, y nos deja añadir al quinto, en este caso añadimos con el número 21, nombre Angel, departamento Realidad virtual, sueldo de 34000, y dirección calle nubes. (Figura 6.1)

**Figura 6.1**



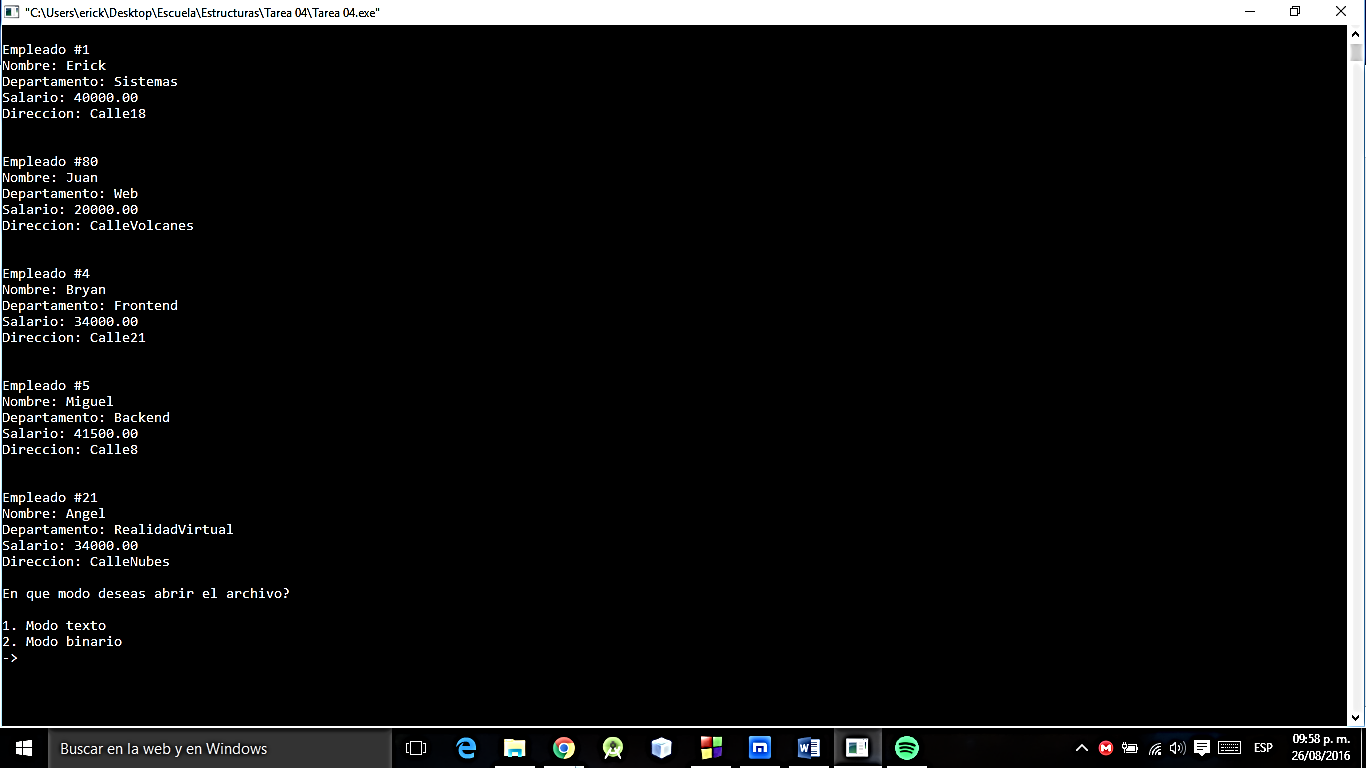
**Figura 6.2**

Como podemos observar, se ha realizado primero el respaldo de lo que se realizó anteriormente (Figura 6.2), y es mostrado una vez realizado, como se ve en la figura 6.3, con los datos que hemos ingresado, en este caso es imposible corroborar como en los casos anteriores, ya que no dispongo de la manera de decodificar el archivo que está en binario.

**Figura 6.3**

## **Prueba 7**

Nuevamente haremos uso de la opción dos, y mostraremos a nuestros usuarios guardados, en modo binario como es evidente. (Figura 7.1)



**Figura 7.1**

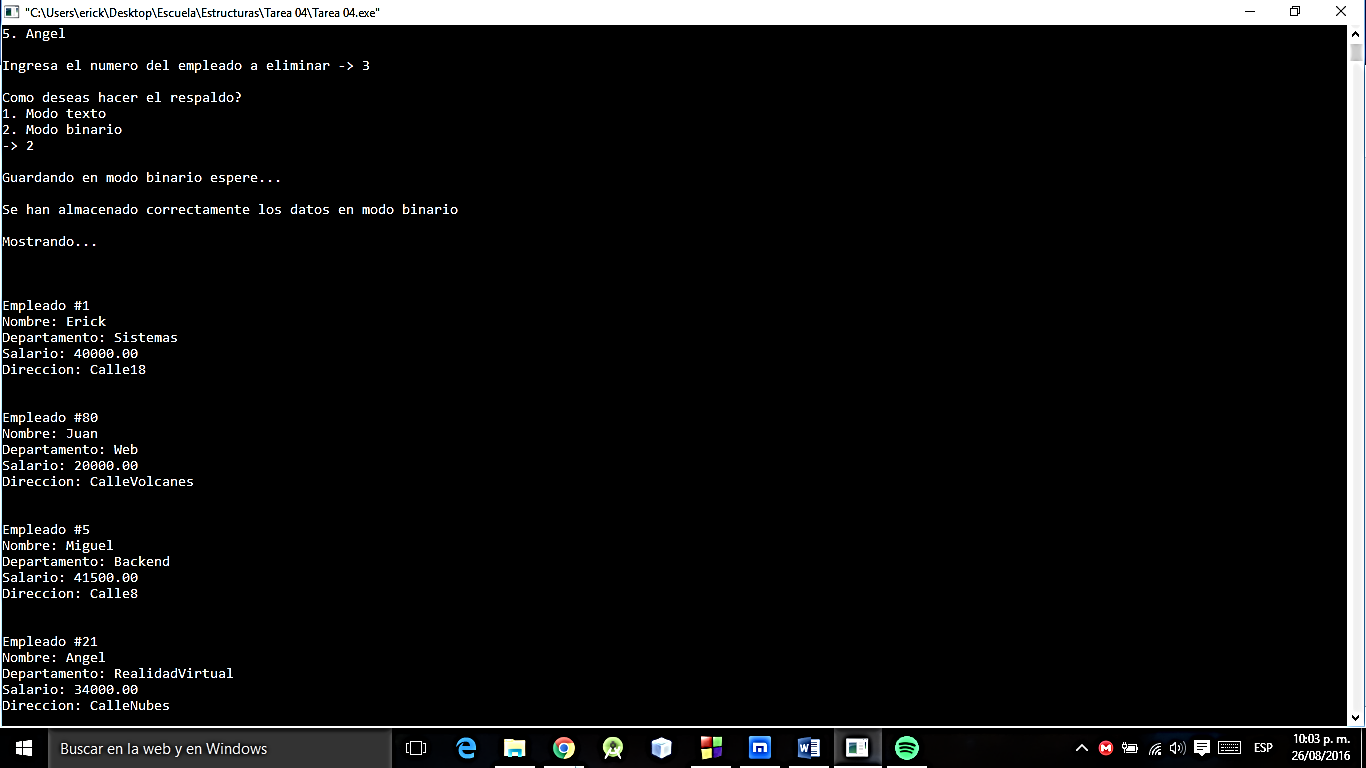
Como podemos observar, en la figura 7.2 se nos hace el muestreo de todo lo que hemos almacenado hast ahora en modo binario, y podemos compararlo con la figura 2.2 y ocurre de la misma forma.

**Figura 7.2**

## **Prueba 8**

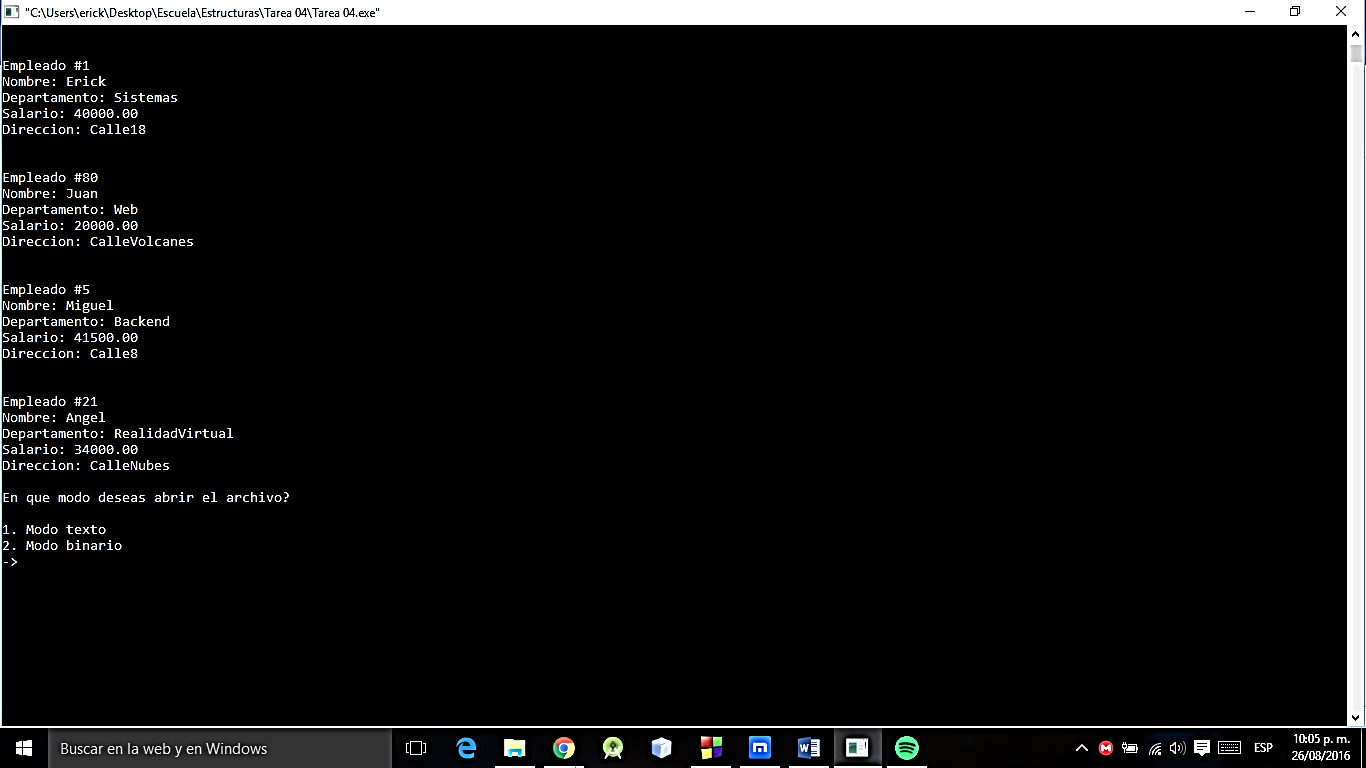
Regresamos al menú, y ahora seleccionamos la opción 5, eliminar empleado, y cómo podemos ver, y como anteriormente habíamos visto, se nos muestran los empleados registrados e ingresamos el que deseamos eliminar. (Figura 8.1)

**Figura 8.1**



Como podemos observar, el usuario que eliminaremos es el 3, y realizamos posteriormente el respaldo en modo binario, se puede apreciar en la figura 8.2

**Figura 8.2**



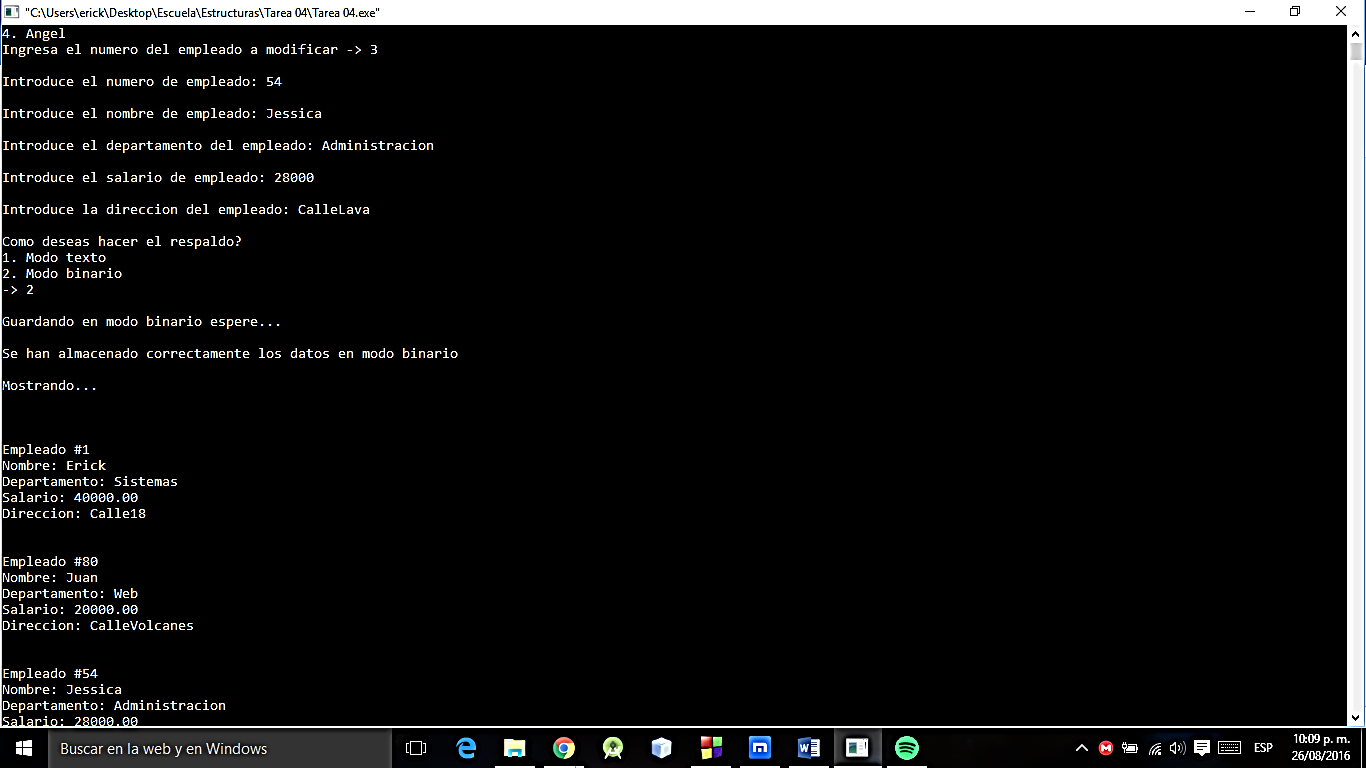
Como podemos observar, en efecto, el empleado Bryan ha sido satisfactoriamente eliminado, y solo nos restan 4 registros. (Figura 8.3)

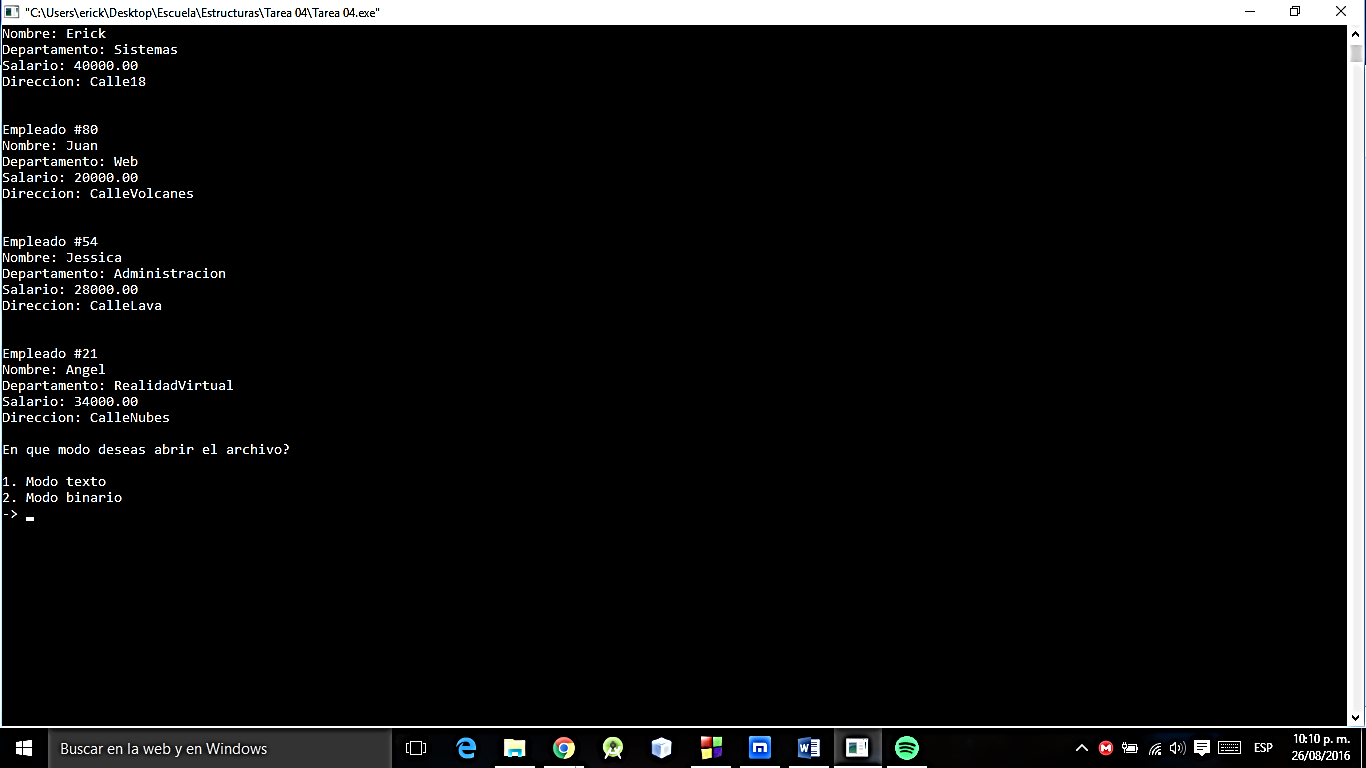
**Figura 8.3**

## **Prueba 9**

Una vez más, nos situamos en el menú, y seleccionamos la opción para hacer modificaciones en los registros, se nos muestran los empleados registrados, en este caso solo 4, y seleccionamos el registro 3 que es el que será modificado. (Figura 9.1)

**Figura 9.1**



En la figura 9.2, observamos que hemos ingresado como numero el 54, como nombre del empleado Jessica, el departamento administración, el salario 28000, y la dirección Calle lava, realizamos el respaldo en modo binario, se hace el respaldo y nos será mostrado.

**Figura 9.2**

Finalmente, se nos muestran los datos, y en efecto, el registro ha sido cambiado satisfactoriamente, y hasta el momento el programa ha funcionado como deseábamos. (Figura 9.3)

**Figura 9.3**

# **Implementación**

/\*

AUTOR: Vargas Romero Erick Efraín

VERSIÓN: 2.0

DESCRIPCIÓN: Modifica el programa del ejemplo para que el programa pueda realizar, altas, bajas

y cambios de los empleados, además de los respaldos en modo texto y binario

según el usuario lo solicite. Crear un menú para operar con el programa. El

programa deberá de ser capaz de salir y almacenar la lista de empleados para que

al iniciar nuevamente mantenga la información de ellos.

OBSERVACIONES: Se definio NUM\_EMP como constante para hacer más compatible el programa

además se definieron las contantes path\_txt(Ruta para el archivo de texto)

y también path\_dat(Ruta para el archivo guardado en binario)

COMPILACIÓN: gcc problema\_empleados.c -o problema\_empleados

EJECUCIÓN: problema\_empleados.exe (En Windows)

NOTA: No fué probado en LINUX

\*/

//LIBRERAS

#include<stdio.h>

#include<windows.h>

//DEFINICIÓN DE ESTRUCTURAS Y CONSTANTES

//Número de empleados

#define NUM\_EMP 100

//Ruta del archívo en modo texto

#define path\_txt "empleados.txt"

//Ruta del rchivo en modo binario

#define path\_dat "empleados.dat"

//Definición de la estructura

**typedef** struct empleado

**{**

int no\_empleado**;**

char nombre**[**45**];**

char departamento**[**45**];**

float salario**;**

char direccion**[**100**];**

**}** empleado**;**

//DECLARACIÓN DE FUNCIONES

/\*Procedimiento para capturar por medio de la entrada estandar los empleados

(Recibe la referencia al arreglo de empleados)\*/

void CapturaEmpleados**(**empleado **\***lista**);**

/\*Procedimiento para mostrar en la salida estandar los empleados

(Recibe la referencia al arreglo de empleados)\*/

void ImprimeEmpleados**(**empleado **\***lista**);**

/\*Procedimiento para guardar en un archivo de texto a los empleados

(Recibe la referencia al arreglo de empleados y la ruta del archivo a escribir)\*/

void GuardaEmpleadosTexto**(**empleado **\***lista**,** char**\*** ruta\_escribir**);**

/\*Procedimiento para leer de un archivo de texto a los empleados

(Recibe la referencia al arreglo de empleados y la ruta del archivo a leer)\*/

void LeeEmpleadosTexto**(**empleado **\***lista**,** char**\*** ruta\_leer**);**

/\*Procedimiento para guardar en un archivo de binario a los empleados

(Recibe la referencia al arreglo de empleados y la ruta del archivo a escribir)\*/

void GuardaEmpleadosBinario**(**empleado **\***lista**,** char**\*** ruta\_escribir**);**

/\*Procedimiento para leer de un archivo binario a los empleados

(Recibe la referencia al arreglo de empleados y la ruta del archivo a leer)\*/

void LeeEmpleadosBinario**(**empleado **\***lista**,** char**\*** ruta\_leer**);**

/\*En esta función, se realizan evaluaciones de lo que

se ingresó en el pequeño menú creado en el main

recibe\_ el arreglo de empleados y un entero para

realizar una seleccion\*/

void selectOptions**(**empleado**\*** lista**,** int option**);**

/\*En esta función elegimos como deseamos leer el archivo

recibe el arreglo de empleados también recibe una variable

tipo int para realizar una selección\*/

void OpenHow**(**empleado **\***lista**,** int show**);**

/\*En esta función hacemos un respaldo de lo que se

ingresó en la estructura, recibe el arreglo de empleados\*/

void CreateBackUp**(**empleado**\*** lista**);**

/\*Esta función verifica si existe o no el archivo

con anterioridad sino es así lo crea, recibe:

un apuntador archivo y la ruta que verificará\*/

void VerifyIfWasCreated**(**FILE**\*** file**,** char**\*** route**);**

/\*Esta función hace la búsqueda de un empleado

y lo elimina primeramente de nuestra estructura

finalmente del fichero creado, recibe:

el arreglo de empleados\*/

void deleteEmployee**(**empleado**\*** lista**);**

/\*Hace un conteo de los empleados que están registrados

haciendo un recorrido por toda la estructura, recibe:

arreglo de empleados además

nos returna un valor de igual manera int\*/

int EmployeesRegistered**(**empleado**\*** lista**);**

/\*Este método nos permite hacer la modificación de algún

usuario, siempre y cuando exista alguno, recibe,

arreglo de empleados\*/

void modifyEmployee**(**empleado**\*** lista**);**

//PROGRAMA PRINCIPAL

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

**{**

empleado lista**[**NUM\_EMP**];**

OpenHow**(**lista**,** 0**);**

system**(**"CLS"**);**

int option**;**

printf**(**"\n\nHola!\nIntroduce la opcion que deseas"**);**

printf**(**"\n1. Capturar empleados\n2. Mostrar empleados\n"**);**

printf**(**"3. Hacer respaldo\n4. Eliminar empleado"**);**

printf**(**"\n5. Modificar empleado\n6. Salir\n-> "**);**

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

selectOptions**(**lista**,** option**);**

**return** 0**;**

**}**

//DEFINICIÃ“N DE FUNCIONES

/\*

void CapturaEmpleados(empleado \*lista);

DescripciÃ³n: Procedimiento para capturar por medio de la entrada estandar los empleados

Recibe: empleado \*lista (Referencia al arreglo de empleados)

Devuelve:

Observaciones: Si la esctructura empleado se modifica, tambiÃ©n se modificarÃ¡ esta funciÃ³n

\*/

void CapturaEmpleados**(**empleado **\***lista**)**

**{**

int i**;**

printf**(**"\nEste programa solo gestiona %d empleados"**,** NUM\_EMP**);**

printf**(**"\nActualmente hay %d empleados registrados"**,** EmployeesRegistered**(**lista**));**

**if((**NUM\_EMP **-** EmployeesRegistered**(**lista**))** **>** 0**)**

**{**

**for(**i**=**0**;** i**<**NUM\_EMP**;** i**++)**

**{**

**if(**isEmpty**(**lista**,** i**)** **==** 0**)**

**{**

printf**(**"\n\n\nEmpleado %d de la lista"**,**i**+**1**);**

printf**(**"\nIntroduce el numero de empleado: "**);**

scanf**(**"%d"**,&**lista**[**i**].**no\_empleado**);**

printf**(**"\nIntroduce el nombre de empleado: "**);**

scanf**(**"%s"**,&**lista**[**i**].**nombre**);**

printf**(**"\nIntroduce el departamento del empleado: "**);**

scanf**(**"%s"**,&**lista**[**i**].**departamento**);**

printf**(**"\nIntroduce el salario de empleado: "**);**

scanf**(**"%f"**,&**lista**[**i**].**salario**);**

printf**(**"\nIntroduce la direccion del empleado: "**);**

scanf**(**"%s"**,&**lista**[**i**].**direccion**);**

**}**

**}**

CreateBackUp**(**lista**);**

**}**

**else**

**{**

printf**(**"\n\nNo es posible guardar mas empleados"**);**

**}**

**}**

/\*

void ImprimeEmpleados(empleado \*lista);

DescripciÃ³n: Procedimiento para mostrar en la salida estandar los empleados

Recibe: empleado \*lista (Referencia al arreglo de empleados)

Devuelve:

Observaciones: Si la esctructura empleado se modifica, tambiÃ©n se modificarÃ¡ esta funciÃ³n

\*/

void ImprimeEmpleados**(**empleado **\***lista**)**

**{**

int i**;**

**for(**i**=**0**;** i**<**NUM\_EMP**;** i**++)**

**{**

**if(**isEmpty**(**lista**,** i**)** **==** 1**)**

**{**

printf**(**"\n\n\nEmpleado #%d"**,**lista**[**i**].**no\_empleado**);**

printf**(**"\nNombre: %s"**,**lista**[**i**].**nombre**);**

printf**(**"\nDepartamento: %s"**,**lista**[**i**].**departamento**);**

printf**(**"\nSalario: %.2f"**,**lista**[**i**].**salario**);**

printf**(**"\nDireccion: %s"**,**lista**[**i**].**direccion**);**

**}**

**}**

**}**

/\*

void GuardaEmpleadosTexto(empleado \*lista, char\* ruta\_escribir);

DescripciÃ³n: Procedimiento para guardar en un archivo de texto a los empleados

Recibe: empleado \*lista (Referencia al arreglo de empleados), char \*ruta(Cadena con la ruta del archivo a escribir)

Devuelve:

Observaciones: Si la esctructura empleado se modifica, tambiÃ©n se modificarÃ¡ esta funciÃ³n, el archivo a escribir se maneja en modo texto.

\*/

void GuardaEmpleadosTexto**(**empleado **\***lista**,** char**\*** ruta\_escribir**)**

**{**

int i**;**

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

remove**(**ruta\_escribir**);**

VerifyIfWasCreated**(**archivo**,** ruta\_escribir**);**

archivo**=**fopen**(**ruta\_escribir**,**"w"**);**

**for(**i**=**0**;** i**<**NUM\_EMP**;** i**++)**

**{**

**if(**isEmpty**(**lista**,** i**)** **==** 1**)**

**{**

fprintf**(**archivo**,**"\n\nEmpleado %d de la lista"**,**i**+**1**);**

fprintf**(**archivo**,**"\nNumero de empleado: %d"**,**lista**[**i**].**no\_empleado**);**

fprintf**(**archivo**,**"\nNombre de empleado: %s"**,**lista**[**i**].**nombre**);**

fprintf**(**archivo**,**"\nDepartamento del empleado: %s"**,**lista**[**i**].**departamento**);**

fprintf**(**archivo**,**"\nSalario de empleado: %f"**,**lista**[**i**].**salario**);**

fprintf**(**archivo**,**"\nDireccion del empleado: %s"**,**lista**[**i**].**direccion**);**

**}**

**}**

fclose**(**archivo**);**

**}**

/\*

void LeeEmpleadosTexto(empleado \*lista, char\* ruta\_escribir);

DescripciÃ³n: Procedimiento para leer de un archivo de texto a los empleados

Recibe: empleado \*lista (Referencia al arreglo de empleados), char \*ruta(Cadena con la ruta del archivo a escribir)

Devuelve:

Observaciones: Si la esctructura empleado se modifica, o si se cambia la la manera de guardar a los empleados,

tambiÃ©n se modificarÃ¡ esta funciÃ³n, el archivo a escribir se maneja en modo texto.

\*/

void LeeEmpleadosTexto**(**empleado **\***lista**,** char**\*** ruta\_escribir**)**

**{**

int i**,**n**;**

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

VerifyIfWasCreated**(**archivo**,** ruta\_escribir**);**

archivo**=**fopen**(**ruta\_escribir**,**"r"**);**

**for(**i**=**0**;** i**<**NUM\_EMP**;** i**++)**

**{**

fscanf**(**archivo**,**"\n\nEmpleado %d de la lista"**,&**n**);**

fscanf**(**archivo**,**"\nNumero de empleado: %d"**,&**lista**[**i**].**no\_empleado**);**

fscanf**(**archivo**,**"\nNombre de empleado: %s"**,&**lista**[**i**].**nombre**);**

fscanf**(**archivo**,**"\nDepartamento del empleado: %s"**,&**lista**[**i**].**departamento**);**

fscanf**(**archivo**,**"\nSalario de empleado: %f"**,&**lista**[**i**].**salario**);**

fscanf**(**archivo**,**"\nDireccion del empleado: %s"**,&**lista**[**i**].**direccion**);**

**}**

fclose**(**archivo**);**

**}**

/\*

void GuardaEmpleadosBinario(empleado \*lista, char\* ruta\_escribir);

DescripciÃ³n: Procedimiento para guardar en un archivo binario a los empleados

Recibe: empleado \*lista (Referencia al arreglo de empleados), char \*ruta(Cadena con la ruta del archivo a escribir)

Devuelve:

Observaciones:

\*/

void GuardaEmpleadosBinario**(**empleado **\***lista**,** char**\*** ruta\_escribir**)**

**{**

int i**;**

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

remove**(**ruta\_escribir**);**

VerifyIfWasCreated**(**archivo**,** ruta\_escribir**);**

archivo**=**fopen**(**ruta\_escribir**,**"wb"**);**

fseek**(**archivo**,**0**,**SEEK\_SET**);**

fwrite**(**lista**,sizeof(**empleado**),**NUM\_EMP**,**archivo**);**

fclose**(**archivo**);**

**}**

/\*

void LeeEmpleadosBinario(empleado \*lista, char\* ruta\_escribir);

DescripciÃ³n: Procedimiento para leer de un archivo binario a los empleados

Recibe: empleado \*lista (Referencia al arreglo de empleados), char \*ruta(Cadena con la ruta del archivo a escribir)

Devuelve:

Observaciones: Si la esctructura empleado se modifica, tambiÃ©n se modificarÃ¡ esta funciÃ³n, el archivo a escribir se maneja en modo binario.

\*/

void LeeEmpleadosBinario**(**empleado **\***lista**,** char**\*** ruta\_escribir**)**

**{**

int i**,**n**;**

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

VerifyIfWasCreated**(**archivo**,** ruta\_escribir**);**

archivo**=**fopen**(**ruta\_escribir**,**"rb"**);**

fread**(**lista**,sizeof(**empleado**),**NUM\_EMP**,**archivo**);**

fclose**(**archivo**);**

**}**

/\*En esta función, se realizan evaluaciones de lo que

se ingresó en el pequeño menú creado en el main\*/

void selectOptions**(**empleado**\*** lista**,** int option**)**

**{**

**switch(**option**)**

**{**

**case** 1**:**

CapturaEmpleados**(**lista**);**

**break;**

**case** 2**:**

OpenHow**(**lista**,** 1**);**

**break;**

**case** 3**:**

CreateBackUp**(**lista**);**

**break;**

**case** 4**:**

deleteEmployee**(**lista**);**

**break;**

**case** 5**:**

modifyEmployee**(**lista**);**

**break;**

**case** 6**:**

exit**(**0**);**

**break;**

**default:**

printf**(**"\nEsa opcion no existe..."**);**

**break;**

**}**

main**();**

**}**

/\*En esta función elegimos como deseamos leer el archivo\*/

void OpenHow**(**empleado **\***lista**,** int show**)**

**{**

int option**;**

cleanStruct**(**lista**);**

printf**(**"\n\nEn que modo deseas abrir el archivo?"**);**

printf**(**"\n\n1. Modo texto\n2. Modo binario\n-> "**);**

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

**switch(**option**)**

**{**

**case** 1**:**

printf**(**"\nAbriendo en modo texto..."**);**

Sleep**(**2000**);**

LeeEmpleadosTexto**(**lista**,** path\_txt**);**

**if(**show **==** 1**)**

**{**

ImprimeEmpleados**(**lista**);**

**}**

**break;**

**case** 2**:**

printf**(**"\nAbriendo en modo binario..."**);**

Sleep**(**2000**);**

LeeEmpleadosBinario**(**lista**,** path\_dat**);**

**if(**show **==** 1**)**

**{**

ImprimeEmpleados**(**lista**);**

**}**

**break;**

**default:**

printf**(**"\nEsa opcion no existe..."**);**

**break;**

**}**

**}**

/\*En esta función hacemos un respaldo de lo que se

ingresó en la estructura\*/

void CreateBackUp**(**empleado**\*** lista**)**

**{**

int option**;**

printf**(**"\nComo deseas hacer el respaldo?"**);**

printf**(**"\n1. Modo texto\n2. Modo binario\n-> "**);**

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

**switch(**option**)**

**{**

**case** 1**:**

printf**(**"\nGuardando en modo texto espere..."**);**

Sleep**(**3000**);**

GuardaEmpleadosTexto**(**lista**,** path\_txt**);**

Sleep**(**500**);**

printf**(**"\n\nSe han almacenado correctamente los datos en modo texto"**);**

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

LeeEmpleadosTexto**(**lista**,** path\_txt**);**

ImprimeEmpleados**(**lista**);**

**break;**

**case** 2**:**

printf**(**"\nGuardando en modo binario espere..."**);**

Sleep**(**3000**);**

GuardaEmpleadosBinario**(**lista**,** path\_dat**);**

Sleep**(**500**);**

printf**(**"\n\nSe han almacenado correctamente los datos en modo binario"**);**

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

LeeEmpleadosBinario**(**lista**,** path\_dat**);**

ImprimeEmpleados**(**lista**);**

**break;**

**default:**

**break;**

**}**

**}**

/\*Esta función verifica si existe o no el archivo

con anterioridad sino es así lo crea\*/

void VerifyIfWasCreated**(**FILE**\*** file**,** char**\*** route**)**

**{**

**if((**file **=** fopen**(**route**,** "r"**))** **==** **NULL)**

**{**

//printf("\nEl archivo no existe creando...");

file **=** fopen**(**route**,** "w"**);**

//Sleep(2000);

//printf("\nEl archivo se ha creado satisfctoriamente...");

//Sleep(2000);

**}**

fclose**(**file**);**

**}**

/\*En esta función, limpiamos la estructura

en todos sus parámetros para que no arroje

ni guarde basura\*/

void cleanStruct**(**empleado**\*** lista**)**

**{**

int i**,** j**;**

**for(**i **=** 0**;** i**<**NUM\_EMP**;** i**++)**

**{**

lista**[**i**].**no\_empleado **=** 0**;**

lista**[**i**].**salario **=** 0**;**

**for(**j **=** 0**;** j**<**45**;** j**++)**

**{**

lista**[**i**].**nombre**[**j**]** **=** '\0'**;**

lista**[**i**].**departamento**[**j**]** **=** '\0'**;**

**}**

**for(**j **=** 0**;** j**<**100**;** j**++)**

**{**

lista**[**i**].**direccion**[**j**]** **=** '\0'**;**

**}**

**}**

**}**

/\*Verifica si lista en cierta posición, en su

parámetro nombre esta o no vacío, si está vacío

la variable flag, se iguala a 0 (falso), sino a

1 (verdadero)\*/

int isEmpty**(**empleado**\*** lista**,** int position**)**

**{**

int flag**;**

**if(**lista**[**position**].**no\_empleado **==** 0 **||** lista**[**position**].**salario **==** 0 **||**

strcmp**(**lista**[**position**].**nombre**,** ""**)** **==** 0 **||**

strcmp**(**lista**[**position**].**direccion**,** ""**)** **==** 0 **||**

strcmp**(**lista**[**position**].**departamento**,** ""**)** **==** 0**)**

**{**

flag **=** 0**;**

**}**

**else**

**{**

flag **=** 1**;**

**}**

**return** flag**;**

**}**

/\*Esta función hace la búsqueda de un empleado

y lo elimina primeramente de nuestra estructura

finalmente del fichero creado\*/

void deleteEmployee**(**empleado**\*** lista**)**

**{**

int i**,** j **=** 1**,** k**,** l **=** **-**1**;**

printf**(**"\nHay %d empleados registrados\n"**,** EmployeesRegistered**(**lista**));**

**if(**EmployeesRegistered**(**lista**)** **>** 0**){**

**for(**i **=** 0**;** i**<**NUM\_EMP**;** i**++)**

**{**

**if(**isEmpty**(**lista**,** i**)** **==** 1**)**

**{**

printf**(**"\n%d. %s\n"**,** j**++,** lista**[**i**].**nombre**);**

**}**

**}**

printf**(**"\nIngresa el numero del empleado a eliminar -> "**);**

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

j**--;**

empleado newList**[**EmployeesRegistered**(**lista**)-**1**];**

**for(**k **=** 0**;** k**<**NUM\_EMP**;** k**++)**

**{**

**if(**k **!=** j**)**

**{**

l**++;**

newList**[**l**].**no\_empleado **=** lista**[**k**].**no\_empleado**;**

newList**[**l**].**salario **=** lista**[**k**].**salario**;**

strcpy**(**newList**[**l**].**nombre**,** lista**[**k**].**nombre**);**

strcpy**(**newList**[**l**].**direccion**,** lista**[**k**].**direccion**);**

strcpy**(**newList**[**l**].**departamento**,** lista**[**k**].**departamento**);**

**}**

**}**

CreateBackUp**(**newList**);**

**}**

**}**

/\*Hace un conteo de los empleados que están registrados

haciendo un recorrido por toda la estructura\*/

int EmployeesRegistered**(**empleado**\*** lista**)**

**{**

int registeredEmp **=** 0**,** i**;**

**for(**i **=** 0**;** i**<**NUM\_EMP**;** i**++)**

**{**

**if(**isEmpty**(**lista**,** i**)** **==** 1**)**

**{**

registeredEmp**++;**

**}**

**}**

**return** registeredEmp**;**

**}**

/\*Este método nos permite hacer la modificación de algún

usuario, siempre y cuando exista alguno\*/

void modifyEmployee**(**empleado**\*** lista**)**

**{**

int i**,** j **=** 1**,** k**;**

printf**(**"\nHay %d empleados registrados\n"**,** EmployeesRegistered**(**lista**));**

**if(**EmployeesRegistered**(**lista**)** **>** 0**){**

**for(**k **=** 0**;** k**<**NUM\_EMP**;** k**++)**

**{**

**if(**isEmpty**(**lista**,** k**)** **==** 1**)**

**{**

printf**(**"\n%d. %s\n"**,** j**++,** lista**[**k**].**nombre**);**

**}**

**}**

printf**(**"Ingresa el numero del empleado a modificar -> "**);**

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

j**--;**

printf**(**"\nIntroduce el numero de empleado: "**);**

scanf**(**"%d"**,&**lista**[**j**].**no\_empleado**);**

printf**(**"\nIntroduce el nombre de empleado: "**);**

scanf**(**"%s"**,&**lista**[**j**].**nombre**);**

printf**(**"\nIntroduce el departamento del empleado: "**);**

scanf**(**"%s"**,&**lista**[**j**].**departamento**);**

printf**(**"\nIntroduce el salario de empleado: "**);**

scanf**(**"%f"**,&**lista**[**j**].**salario**);**

printf**(**"\nIntroduce la direccion del empleado: "**);**

scanf**(**"%s"**,&**lista**[**j**].**direccion**);**

CreateBackUp**(**lista**);**

**}**

**}**