Universidad Autónoma de Baja California

Facultad de Ingeniería Arquitectura y Diseño

Ingeniero en Software y tecnologías emergentes

Materia: Programación Estructurada / Clave **36276**

Alumno: Fernando Haro Calvo

Matrícula: 372106

Maestro: Pedro Núñez Yépiz

Actividad No. 12

Tema - Unidad 1: Archivos de Texto

Ensenada Baja California a 12 de noviembre del 2022

Universidad Autónoma de Baja California

Facultad de Ingeniería Arquitectura y Diseño

1. INTRODUCCIÓN

Esta práctica es una combinación de los conocimientos adquiridos de la práctica 10 más nuestros conocimientos acerca de lectura/escritura de archivos de texto. Consiste en generar aleatoriamente información para rellenar un vector de datos de personas, donde también podremos realizar métodos de búsqueda y ordenación, al igual que imprimir los registros en la consola en forma de tabla o de registro y poder generar un archivo de texto con estos mismos. Además, el programa deberá mantener registro de las personas en el vector, ya sean activos o inactivos en diferentes archivos.

2. COMPETENCIA

El objetivo principal poner en práctica los conocimientos adquiridos en programación en C, especialmente abordará:

Estructuras de dato 'struct': Se utiliza para almacenar diferentes tipos de datos en una sola entidad.

Algoritmos de búsqueda (secuencial y binaria): se refiere a la capacidad de diseñar e implementar algoritmos que permitan encontrar un elemento específico en una colección de datos, como un arreglo.

Validación de Datos: El programa debe ser capaz de validar los datos de entrada, como el nombre, fecha de nacimiento, sexo y entidad federativa, para asegurarse de que sean correctos y cumplan con los requisitos oficiales.

Modularidad y Organización: La práctica debe demostrar una estructura modular y organizada del código, utilizando funciones y procedimientos para dividir el problema en tareas más pequeñas y manejables.

Excepciones y Control de Errores: El programa debe ser robusto, manejando excepciones y errores de manera adecuada. Debe ser capaz de informar al usuario si se ingresan datos incorrectos y nunca terminar inesperadamente.

Archivos de Texto: El programa deberá aplicar las estrategias de escribir, anidar y leer archivos de texto en formato ".txt" para almacenar información.

3. FUNDAMENTOS

Programación en C: La práctica se basa en la programación en el lenguaje C, que es ampliamente utilizado en el desarrollo de sistemas y aplicaciones de software.

Funciones: Las funciones son bloques de código reutilizables que realizan tareas específicas. La práctica se enfoca en la creación y uso de funciones para organizar y modularizar el código.

Structs: Son una colección de variables llamadas "miembros" que pueden tener diferentes tipos de datos, como enteros, caracteres, flotantes u otros structs. Los structs se utilizan para representar una entidad o conjunto de datos que tiene múltiples atributos relacionados.

Cadenas: Son una secuencia de caracteres, es decir, una serie de caracteres que se organizan de manera consecutiva. Estos caracteres pueden ser letras, números, símbolos y espacios en blanco.

Validación: Es una parte esencial de la programación para garantizar que los datos sean correctos y seguros, y para evitar errores o comportamientos inesperados en el programa.

Generación de archivos ".txt": Utilizando los datos de un struct, generar una archivo de texto usando las funciones de la librería estándar de C.

4. PROCEDIMIENTO

MENÚ

1.- Cargar Archivo
2.- Agregar
3.- Eliminar
4.- Buscar
5.- Ordenar
6.- Mostrar Todo
7.- Generar Archivo
8.- Cantidad de registros en Archivo
9.- Mostrar Borrados
0.- Salir

INSTRUCCIONES: Programa que contenga el menú anterior, el programa utiliza un vector de registros de máximo 1,500 registros, de la siguiente estructura: **[status, matricula, ApPat, ApMat, Nombre, Edad, sexo]** donde el campo llave es matricula**.**

datos.txt es el archivo con los registros a cargar en el vector de registros

1.- Cargar Archivo: El programa deberá cargar el vector de registros desde el archivo de texto (solo podrá cargarse una sola vez el archivo)

2.- Agregar: El programa deberá ser capaz de agregar 10 registros al arreglo y al final del archivo de texto. (Generar automáticamente los datos).

3.- Eliminar: El programa deberá buscar una matrícula en el vector por medio del método de búsqueda más óptimo. Utilizar banderas para escoger el método más adecuado., imprimir el registro y preguntar si se quiere eliminar el registro, (al cerrar el programa se deberá agregar al archivo borrados el registro o registros eliminados, así se deberá mantener dos archivos uno con datos válidos y otro con los datos que se borraron)

4.- Buscar: El programa deberá buscar una matrícula en el vector por medio del método de búsqueda más óptimo. Utilizar banderas para escoger el método más adecuado. Mostrar los datos en forma de registro

5.- Ordenar: El programa deberá ordenar el vector por medio del método de ordenación más óptimo. Utilizar banderas para escoger el método más adecuado se ordenará por el campo llave (matrícula)

6.- Mostrar Todo: El programa deberá mostrar todos los registros del vector tal y como están en ese momento ordenado o desordenado. (mostrar en forma de tabla, y de 40 en 40)

7.- Generar Archivo: El programa deberá preguntar al usuario el nombre del archivo, solo nombre sin extensión, el programa generará un archivo con el nombre proporcionado por el usuario con extensión .txt los datos que pondrá en el archivo de texto serán idénticos a los contenidos en el Vector de registros. (ordenado o desordenado). El programa podrá generar múltiples archivos para comprobar las salidas.

8.- Cantidad de registros en archivo: El programa deberá llamar a un archivo externo, donde mande ejecutar el archivo y como parámetros el nombre del archivo que se desea evaluar, el programa externo deberá ser capaz de retornar un valor enero que sea la cantidad de registros que contiene el archivo en cuestión

9.- Mostrar Borrados: El programa deberá mostrar el archivo de texto tal y como se visualiza con la cantidad de registros que se eliminaron del archivo original y que fueron marcados en su momento como registros eliminados

NOTA: Programa 100% Validado.

NOTA 2: Si el programa vector cambia de tamaño por agregar un nuevo registro al salir el programa deberá actualizar o sustituir el **datos.txt (deberá contener puros registros válidos los eliminados o inactivos deberán ser parte del archivo de texto2)**

NOTA 3: Usar librería propia

NOTA 4: archivo.txt es un ejemplo de cómo debe ser el archivo que se genere

5. RESULTADOS Y CONCLUSIONES

La actividad me sirvió para reforzar y poner en práctica lo aprendido acerca de archivos de texto, ayudándome a dominarlos mucho más, fue un poco más sencilla esta práctica comparándola con pasadas, debido a que se simplificó la estructura de Tprogra (persona), y además no se requirió la validación manual de datos como en el caso de la CURP.

Uno de los aspectos más interesantes de la práctica fue actualizar un archivo de texto y acceder a este mediante otro archivo ".c", como se realizó en el caso de la función de contarRegArch:

```
int main(int argc, char *argv[])
{
    if (argc != 3)
    {
        printf("Error en los argumentos\n");
        system("pause");
        return -1;
    }

    FILE *fa;
    char nomArchivo[30];
    char linea[90];
    int cont = 0;
    int status = atoi(argv[2]);
    strcpy(nomArchivo, argv[1]);
```

En resumen, la práctica me ayudó a lograr un manejo más eficiente de archivos, vectores y métodos de búsqueda y ordenación, cumpliendo con las recomendaciones establecidas.

6. ANEXOS

Código y salidas: HCF_ACT12_ANEXOS
Archivo: HCF_ACT12_932.c
Librería: alexandria.h

7. REFERENCIAS

Diseño de algoritmos y su codificación en lenguaje C
Corona, M.A. y Ancona, M.A. (2011)..
España: McGraw-Hill.
ISBN: 9786071505712

Programación estructurada a fondo:implementación de algoritmos en C
:Pearson Educación.Sznajdleder, P. A. (2017)..
Buenos Aires,Argentina: Alfaomega

Como programar en C/C++
H.M. Deitel/ P.J. Deitel
Segunda edición
Editorial: Prentice Hall.
ISBN:9688804711

Programación en C.Metodología, estructura de datos y objetos
Joyanes, L. y Zahonero, I. (2001)..
España:McGraw-Hill.
ISBN: 8448130138