Universidad Autónoma de Baja California

Facultad de Ingeniería Arquitectura y Diseño

Ingeniero en Software y tecnologías emergentes

Materia: Programación Estructurada / Clave **36276**

Alumno: Antonio Ramos González

Matrícula: 372576

Maestro: Pedro Núñez Yépiz

Actividad No. : 12

Tema - Unidad : Archivos

Ensenada Baja California a 12 de noviembre del 2023

1. INTRODUCCIÓN

La gestión de archivos es una parte fundamental en lo que respecta al almacenamiento de datos. Dichos archivos nos sirven para guardar todo tipo de registros y poder manipularlos cuantas veces queramos.

2. COMPETENCIA

En esta práctica se busca que el alumno comprenda de manera sólida la manipulación de archivos de texto, logrando leer y escribir en dichos archivos. También se busca reforzar temas vistos en prácticas anteriores como lo son los métodos de ordenación y búsqueda, validaciones, uso de funciones y el uso de nuestra librería propia.

3. FUNDAMENTOS

Los archivos son un método para el almacenamiento de datos de una forma más o menos permanente. Para acceder a los archivos se debe generar una variable puntero FILE

FILE *fa;

Posteriormente se abrirá el archivo con la función fopen, donde entre parentecis pondremos la ubicación del archivo y separado por una coma y entre comillas colocaremos:

R=lectura del archivo

W=escritura del archivo

A=añadir a archivo

R+=lectura y escritura

W+= escritura y lectura

A+=añadir lectura y escritura

fa=fopen (ubicación archivo, “r”);

así como el archivo debe iniciarse o abrirse, también debe cerrarse usando la función

fclose(fa);

para la manipulación de datos dentro de un archivo se usa fscanf para la lectura y fprintf para la escritura, primero inicializándose el archivo y posteriormente escribiendo o leyendo en el

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 un 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 deberar agregar al archivo borrados el registro o registros eliminados, asi se debera mantener dos archivos uno con datos validos 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 parametros el nombre del archivo que se desea evaluar, el programa externo deberar ser capaz de retornar un valor enero que sea la cantida de registros que contiene el archivo en cuestion

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 momnto como registros eliminados

5. RESULTADOS Y CONCLUSIONES

El trabajar con archivos ofrece una manera eficiente el manejo y almacenamiento de datos, creando programas los cuales nos permiten enviar y recibir datos de dichos archivos para su posterior manipulación.

```
// Antonio Ramos Gonzalez Mt: 372576
// 01/11/2023
// Se generara un archivo donde se almacenara datos generados por el usuario
// RGA_Act12_932
#include "Babilonia.h"
#define N 2000
typedef struct _Reg···
} Treg;

int msg();
void menu();
int verMt(Treg reg[], int n, int mt);
int cargar(Treg reg[], int n, char filename[]);
Treg agregar(Treg reg[], int i);
void nombreAl(char nombre[], int sex);
void apAl(char apellido[]);
int searchSec(Treg reg[], int n, int mt);
int searchBin(Treg reg[], int inf, int sup, int mt);
int orderBu(Treg reg[], int n);
void pintReg(Treg reg[], int n);
void pintOneReg(Treg reg[], int n);
void swap(Treg reg[], int i, int j);
void createTXT(Treg reg[], int n);
int partition(Treg reg[], int low, int high);
void quicksort(Treg reg[], int low, int high);
void RegEnArch();
void regEliminados(Treg reg[], int n);

int main()···

int msg()···

void menu()···

void nombreAl(char nombre[], int sex)···

void apAl(char apellido[])···

Treg agregar(Treg reg[], int i)···

int searchSec(Treg reg[], int n, int mt)···

int searchBin(Treg reg[], int inf, int sup, int mt)···

int orderBu(Treg reg[], int n)···

void pintReg(Treg reg[], int n)···

void pintOneReg(Treg reg[], int n)···

int verMt(Treg reg[], int n, int mt)···

void createTXT(Treg reg[], int n)···

void swap(Treg reg[], int i, int j)···

int partition(Treg reg[], int low, int high)···

void quicksort(Treg reg[], int low, int high)···

int cargar(Treg reg[], int n, char filename[])···

void RegEnArch()···

void regEliminados(Treg reg[], int n)···
```

6. ANEXOS

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