Universidad Autónoma de Baja California

Facultad de Ingeniería Arquitectura y Diseño

Ingeniero en Software y tecnologías emergentes

Materia: Programación Estructurada / Clave **36276**

Alumno: Teresa Rivas Gómez

Matrícula: 372565

Maestro: Pedro Núñez Yépiz

Actividad No. : 13

Tema - Unidad : ARCHIVOS BINARIOS

Ensenada, Baja California a 19 de Noviembre del 2023

Universidad Autónoma de Baja California

Facultad de Ingeniería Arquitectura y Diseño

Introducción:

En esta actividad estuvimos manejando archivos de texto, desde cargar un archivo creado previamente, hasta crear nosotros nuestros propios archivos ".txt". Estos generándolos con ayuda de funciones como "generar_random( )", realizadas previamente en actividades anteriores. Usamos estructuras, librerías propias, cadenas, etc.

Competencias:

De esta actividad se espera hacer buen uso de los archivos de texto, de continuar aprendiendo a usar de manera correcta el bash, las funciones y librerías de C.

Documentación:

Archivos anexados en actividad y apuntes de clases anteriores, la mayoría del código lo pude realizar con el material ya hecho en la Actividad 11 y 12.

Actividad:

ACTIVIDAD 13

REALICE EL SIGUIENTE PROGRAMA QUE CONTENGA UN MENÚ.

MENÚ
1.- AGREGAR (AUTOM 100 REGISTROS)
2.- EDITAR REGISTRO
3.- ELIMINAR REGISTRO (lógico)
4.- BUSCAR
5.- ORDENAR
6- IMPRIMIR
7.- GENERAR ARCHIVO TEXTO
8.- VER ARCHIVO TEXTO
9.- CREAR ARCH BINARIO
10.- CARGAR ARCH BINARIO
11.- MOSTRAR ELIMINADOS
0.- SALIR

UTILIZAR UN ARREGLO DE 5000 REGISTROS. SE DEBERÁ UTILIZAR ESTRUCTURAS CON LOS DATOS BÁSICOS DE UN EMPLEADO.
Preguntar nombre de archivo binario o de archivo texto

Universidad Autónoma de Baja California

Facultad de Ingeniería Arquitectura y Diseño

Busqueda y Ordenacion por CAMPO LLAVE

nota: usar librería propia con funciones
nota2: 100 % validado, Cuidar desbordamiento de vector
nota3: Campo llave matricula no repetido, archivos solo cargar 1 sola vez.
nota4: Usar el tipo Tkey para hacer mas practico el programa

INSTRUCCIONES DEL MENU

1.- Agregar : El programa deberá ser capaz de agregar 100 registros al vector de registros **(Generar automáticamente los datos)**.

2.- Editar Registro : El programa deberá buscar una matrícula en el vector por medio del método de búsqueda más óptimo. **Mostrar los datos en forma de registro** Preguntar que campo quire Editar, actualiar los datos en el vector (solo a registros activos)

3.- Eliminar Registro : 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.

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.- Imprimir: El programa deberá mostrar todos los registros del vector y como están en ese momento ordenado o desordenado. **(mostrar en forma de tabla)**

7.- Generar Archivo Texto : 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.- Mostrar Archivo Texto: 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** mostrar el archivo de texto tal y como se encuantra.

9.- Crear archivo binario : El programa deberá crear un archivo binario con los datos del vector actualizados, sustituir el archivo base, realizar respaldo del archivo anterior y guardarlo con el mismo nombre pero extencion .tmp (validar msges si el archivo no se puede crear por falta de registros en el vector)

10 .- Cargar Archivo Binario : El programa deberá cargar al vector los registros del archivo binario **(solo podrá cargarse una sola vez el archivo, el archivo binario se debara llamar datos.dll y si no existe debera indicar)**

11.- Mostrar Borrados: El programa deberá mostrar del archivo binario solo los registros que se eliminaron (marcados con status 0) y que fueron marcados en su momento como registros eliminados.

Partes importantes en mi programa:

Funciones nuevas para el menu principal que no se repitieron de la actividad pasada:

```
/* EDITAR */
Tdatos modificar_registros(Tdatos almacen[], int i)
{
    int dato;
    dato = ValidarCadena("QUE DATO DESEAS EDITAR?\n[1]NOMBRE\n[2]PRIMER APELLIDO\n[3]SEGUNDO
APELLIDO\n[4]EDAD\n[5]PUESTO\n", 1, 5);
    switch (dato)
    {
```

```
            case 1:
                printf("INGRESA EL NUEVO NOMBRE: \n");
            ciclo(almacen[i].nombre);
            break;
        case 2:
            printf("INGRESA EL NUEVO PRIMER APELLIDO: \n");
            ciclo(almacen[i].apellido1);
            break;
        case 3:
            printf("INGRESA EL NUEVO SEGUNDO APELLIDO: \n");
            ciclo(almacen[i].apellido2);
            break;
        case 4:
            printf("INGRESA LA NUEVA EDAD: \n");
            almacen[i].edad = ValidarCadena("INGRESA LA NUEVA EDAD: \n", 18, 100);
            break;
        case 5:
            printf("INGRESA EL NUEVO PUESTO: \n");
            ciclo(almacen[i].puesto);
            break;
    }
    return almacen[i];
}

/* VER */
void ver_registro(char cadena[])
{
    char nombre_direccion[1000];
    char caracter;
    strcpy(nombre_direccion, "C:\\Users\\52616\\OneDrive\\Documentos\\U A B C\\3er
Semestre\\Programacion Estructurada\\Actividad 13\\output\\");
    strcat(nombre_direccion, cadena);
    strcat(nombre_direccion, ".txt");
    printf("%s\n", nombre_direccion);
    FILE *archivo = fopen(nombre_direccion, "r");
    if (archivo == NULL)
    {
        printf("NO SE PUDO ABRIR EL ARCHIVO\n");
        return;
    }
    do {
        caracter = fgetc(archivo);
        printf("%c", caracter);
    } while (!feof(archivo));
    fclose(archivo);
}
/* CREAR */
/* Crea un archivo binario y escribe en el los datos almacenados en un array de estructuras
(almacen). Como
parametros tenemos el array almacen para los datos almacenados, i para la cantidad y una cadena
para escribir
el nombre del archivo binario que se va a crear. Se abre con el modo de escritura binaria que es
"wb". */
int crear_binario(Tdatos almacen[], int i, char cadena[])
{
```

```
        int j;
        Tdatos _datos;
    char nombre_direccion[1000];
    strcpy(nombre_direccion, "C:\\Users\\52616\\OneDrive\\Documentos\\U A B C\\3er
Semestre\\Programacion Estructurada\\Actividad 13\\output\\");
    strcat(nombre_direccion, cadena);
    strcat(nombre_direccion, ".dll");
    FILE *archivo_binario;
    archivo_binario = fopen(nombre_direccion, "wb");
    for (j = 0; j < i; j++)
    {
        _datos = almacen[j];
        fwrite(&_datos, sizeof(Tdatos), 1, archivo_binario);



}
    fclose(archivo_binario);
    return 0;
}
/* CARGAR */
/* Carga datos desde un archivo binario y los almacena en un array de estructuras (almacen). Tiene
los mismos parametros
que la funcion anterior pero aqui añadi un entero "max" para la cantidad maxima que tiene el array
y se abre el archivo
binario en modo de lectura "lb", para despues almacenarlos en el array (almacen). */
int cargar_binario(Tdatos almacen[], int i, int max, char cadena[])
{
    Tdatos _datos;
    FILE *binario;
    binario = fopen(cadena, "rb");
    if (binario)
    {
        while (fread(&_datos, sizeof(Tdatos), 1, binario))
        {
            if (i < max)
            {
                almacen[i] = _datos;
                i++;
            }
        }
        fclose(binario);
        printf("ARCHIVO ABIERTO CORRECTAMENTE\n");
        return i;
    }
    else
    {
        printf("NO SE PUDO ABRIR EL ARCHIVO\n");
        return -1;
    }
}
```

Estructura modificada para usar las llaves:

```
/*  DECLARAR LLAVE */
typedef int Tllave;
```

```c
/* ESTRUCTURA PRINCIPAL*/
typedef struct _datos{
    int status;
    Tllave matricula;
    char puesto[20];
    char apellido1[20];
    char apellido2[20];
    char nombre[20];
    char telefono[MAX_TELEFONO + 1];
    int edad;
    int sex;
    char sexo[10];
} Tdatos;
```

Respaldo de registros para el archivo binario:

```c
/* RESPALDO BINARIO */
/* Crea un respaldo de un archivo binario. Abre el archivo original y de respaldo, utiliza un buffer para copiar datos en bloques. */
int respaldo_binario( char cadena[])
{
    char direccion_original[1000];
    strcpy(direccion_original, "C:\\Users\\52616\\OneDrive\\Documentos\\U A B C\\3er Semestre\\Programacion Estructurada\\Actividad 13\\output\\");
    strcat(direccion_original, cadena);
    strcat(direccion_original, ".dll");
    char direccion_respaldo[1000];
    strcpy(direccion_respaldo, "C:\\Users\\52616\\OneDrive\\Documentos\\U A B C\\3er Semestre\\Programacion Estructurada\\Actividad 13\\output\\");
    strcat(direccion_respaldo, cadena);
    strcat(direccion_respaldo, ".tmp");
    FILE *origen = fopen(direccion_original, "rb");
    if (origen == NULL)
    {
        printf("ERROR AL ABRIR EL ARCHIVO ORIGINAL");
        fclose(origen);
        return -1;
    }
    FILE *respaldo = fopen(direccion_respaldo, "wb");
    if (respaldo == NULL)
    {
        printf("ERROR AL ABRIR EL ARCHIVO DE RESPALDO");
        fclose(origen);
        return -1;
    }
    char buffer[TAM_BLOQUE];
    size_t bytes_leidos;
    while ((bytes_leidos = fread(buffer, 1, TAM_BLOQUE, origen)) > 0)
    {
        fwrite(buffer, 1, bytes_leidos, respaldo);
    }
    printf("ARCHIVO DE RESPALDO CREADO CORRECTAMENTE\n");
```

```
    fclose(origen);
    fclose(respaldo);
    return 0;
}
```