

TRABAJO OBLIGATORIO DE ESTRUCTURAS DE DATOS Y ALGORITMOS

EDITOR DE TEXTO SIMPLE - PRIMERA PARTE

El obligatorio consiste en la construcción de un sistema para modelar un editor de texto simple que permita trabajar con archivos de texto básicos.

CARACTERÍSTICAS GENERALES - PRIMERA PARTE:

1. El editor trabajará con un archivo de texto simple, sin versiones.
2. El archivo contendrá una lista de líneas de texto que pueden ser modificadas.
3. Las líneas se numeran secuencialmente desde 1 hasta n (donde n es el número total de líneas).
4. Se pueden insertar líneas en cualquier posición válida.
5. Se pueden eliminar líneas de cualquier posición válida.
6. En el sistema se distingue el uso de minúsculas y mayúsculas. Por ejemplo CASA $\neq$ Casa $\neq$ casa.

EJEMPLO DE USO:

Un archivo de texto simple puede contener líneas como:

```
1 Nombre: Juan Pérez
2 Dirección: Rivera 1234
3 Teléfono: 6111111
4 Estado Civil: Soltero
```

Este archivo puede ser modificado insertando nuevas líneas o eliminando líneas existentes.

TIPOS DE DATOS A MANEJAR:

```
TipoRet enum _retorno{
    OK, ERROR, NO_IMPLEMENTADA
};

typedef enum _retorno TipoRet;

// Estructura para una línea de texto
struct _linea {
    char * contenido;          // Contenido de la línea
    struct _linea * siguiente; // Puntero a la siguiente línea
};
typedef struct _linea * Linea;

// Estructura principal del archivo
struct _archivo {
    char * nombre;          // Nombre del archivo
    Linea primera_linea;    // Puntero a la primera línea
    unsigned int cantidad_lineas; // Contador de líneas
};
typedef struct _archivo* Archivo;
```

Pueden definirse tipos de datos (estructuras de datos) auxiliares.

ESTRUCTURA DE DATOS SUGERIDA:

La estructura propuesta utiliza una lista enlazada simple para almacenar las líneas del archivo:

- struct _linea: Representa una línea individual con su contenido y puntero a la siguiente línea
- struct _archivo: Contiene el nombre del archivo, puntero a la primera línea y contador de líneas

Esta estructura permite:

- Insertar líneas en cualquier posición
- Eliminar líneas de cualquier posición
- Recorrer todas las líneas para mostrarlas
- Mantener un contador de líneas para validaciones

Nota: Los estudiantes pueden implementar variaciones de esta estructura según su preferencia, siempre que cumplan con los requisitos funcionales.

El sistema debe permitir realizar las operaciones que se detallan a continuación. Cada operación, excepto CrearArchivo, puede ser ejecutada exitosamente, retornando OK, ó puede generar un error, retornando ERROR, e imprimiendo: ERROR: mensaje. Los mensajes posibles para cada operación del sistema serán indicados. En caso de producirse un error, la estructura que representa a las versiones de un archivo (de tipo Archivo) permanecerá inalterada.

En la descripción de cada operación se incluye al menos un ejemplo. **En todas las operaciones, excepto en la primera, se retoma el resultado del ejemplo del caso anterior.**

OPERACIONES RELATIVAS A LA CREACIÓN Y DESTRUCCIÓN DE UN ARCHIVO:

1) Crear el archivo.

```
Archivo CrearArchivo(char * nombre);
```

Crea el archivo con el nombre especificado y lo inicializa sin contenido (vacío). El archivo creado es retornado. Esta operación se ejecuta al inicio de una sesión de trabajo con un archivo.

Ejemplo:

```
Archivo a = CrearArchivo("curriculum.txt");  
MostrarTexto(a);
```

Salida:

```
curriculum.txt  
  
No contiene líneas
```

Nota: MostrarTexto se explica más adelante.

2) Borrar el archivo.

```
TipoRet BorrarArchivo(Archivo &a);
```

Elimina toda la memoria utilizada por el archivo y asigna NULL al puntero a. Se asume como precondition que a referencia a un archivo (en particular a es distinto a NULL). Esta operación se ejecuta al final de una sesión de trabajo con un archivo.

Retornos posibles:

- **OK** Siempre retorna OK.
- **ERROR** No existen errores posibles.
- **NO_IMPLEMENTADA** Cuando aún no se implementó. Es el tipo de retorno por defecto.

OPERACIONES DEL EDITOR DE TEXTO:

1) Insertar una línea de texto en cierta posición del archivo.

```
TipoRet InsertarLinea(Archivo &a, char * linea, unsigned int nroLinea);
```

Esta función inserta una línea de texto en el archivo en la posición nroLinea. El número de línea debe estar entre 1 y n+1, siendo n la cantidad de líneas del archivo. Por ejemplo, si el texto tiene 7 líneas, se podrá insertar en las posiciones 1 (al comienzo) a 8 (al final). Si se inserta en un número de línea existente, ésta y las siguientes líneas se correrán hacia adelante (abajo) dejando el espacio para la nueva línea.

Ejemplo:

```
Archivo a = CrearArchivo("curriculum.txt");
InsertarLinea(a, "Dirección: Rivera 1234", 1);
InsertarLinea(a, "Teléfono: 6111111", 2);
InsertarLinea(a, "Nombre: Juan Pérez", 1);
InsertarLinea(a, "Estado Civil: Soltero", 4);
MostrarTexto(a);
```

Salida:

```
curriculum.txt

1 Nombre: Juan Pérez
2 Dirección: Rivera 1234
3 Teléfono: 6111111
4 Estado Civil: Soltero
```

Retornos posibles:

- **OK** Si se pudo insertar la línea en la posición especificada del archivo.
- **ERROR** Si nroLinea no es válido.
- **NO_IMPLEMENTADA** Cuando aún no se implementó. Es el tipo de retorno por defecto.

2) Borrar una línea de texto en cierta posición del archivo.

```
TipoRet BorrarLinea(Archivo &a, unsigned int nroLinea);
```

Esta función elimina una línea de texto del archivo en la posición nroLinea. El número de línea debe estar entre 1 y n, siendo n la cantidad de líneas del archivo. Por ejemplo, si el texto tiene 7 líneas, se podrán eliminar líneas de las posiciones 1 a 7. Cuando se elimina una línea, las siguientes líneas se corren, decrementando en una unidad sus posiciones para ocupar el lugar de la línea borrada.

Ejemplo:

```
BorrarLinea(a, 3);
BorrarLinea(a, 3);
MostrarTexto(a);
```

Salida:

```
curriculum.txt

1 Nombre: Juan Pérez
2 Dirección: Rivera 1234
```

Retornos posibles:

- **OK** Si se pudo eliminar la línea con éxito
- **ERROR** Si nroLinea no es válido.
- **NO_IMPLEMENTADA** Cuando aún no se implementó. Es el tipo de retorno por defecto.

3) Mostrar el texto completo del archivo.

```
TipoRet  MostrarTexto(Archivo a);
```

Esta función muestra el texto completo del archivo.

FORMATO: En primer lugar se muestra el nombre del archivo. Después de una línea en blanco lista todas las líneas del texto. Cada línea comienza con el número de línea y separado por un tabulador se mostrará el texto.

Si el archivo no contiene líneas se mostrará la siguiente salida.

Ejemplo:

```
Archivo a = CrearArchivo("curriculum.txt");  
MostrarTexto(a);
```

Salida:

```
curriculum.txt  
  
No contiene líneas
```

Retornos posibles:

- **OK** Si se pudo mostrar el texto, aún cuando éste no contenga líneas.
- **ERROR** No existen errores posibles.
- **NO_IMPLEMENTADA** Cuando aún no se implementó. Es el tipo de retorno por defecto.

CATEGORÍA DE OPERACIONES - PRIMERA PARTE

TIPO 1	Operaciones imprescindibles para que el trabajo obligatorio sea corregido.
TIPO 2	Operaciones importantes. Estas serán probadas independientemente, siempre que estén correctamente implementadas las operaciones de TIPO 1.

TIPO 1	TIPO 2
1) CrearArchivo	2) BorrarArchivo
1) InsertarLinea	2) BorrarLinea
3) MostrarTexto	