Manejador de Dataframes

PRÁCTICA DE FUNDAMENTOS DE PROGRAMACIÓN

Se pide desarrollar un programa en C que, a través de una interfaz de línea de comandos (CLI, "*Command Line Interface*"), gestione un conjunto de datos estructurados en forma de "dataframe", (similar a un dataframe de Python). El objetivo de la práctica es que los alumnos implementen a bajo nivel la estructura y las operaciones básicas para manejar datos organizados en filas y columnas, soportando diferentes tipos de datos (texto, numéricos y fechas), y detectando valores nulos o anómalos.

El programa debe permitir cargar datos de ficheros CSV, mostrando en la interfaz CLI los errores que se detecten, como valores de tipo incorrecto o nulos en las columnas. Una vez cargado algún CSV, se podrán realizar otras operaciones sobre las filas y/o columnas del "dataframe" como obtener estadísticas básicas, filtrar filas según condiciones específicas, generar nuevas columnas derivadas, etc...

El programa se ejecutará, de manera general, siguiendo el flujo de control que se indica a continuación:

1	**Imprimir en pantalla los datos del alumno:** **(nombre, apellidos y correo electrónico)**
2	**Mostrar el prompt en color blanco**
3	**Leer el comando introducido por el usuario por teclado**
4	**Analizar el comando del usuario:**
5	**⇒ Si el comando es incorrecto o no válido se indica con un mensaje de error en rojo y se vuelve al paso 2**
6	**⇒ Si el comando es nulo o vacío no se hace nada y se vuelve al paso 2**
7	**⇒ Si el comando es "quit" se libera toda la memoria dinámica que el programa esté ocupando y se finaliza mostrando por pantalla el mensaje "Fin..." en verde**
8	**⇒ Si el comando el válido, se ejecuta la operación correspondiente y, si es necesario, se muestra el resultado por pantalla en color verde. Al terminar se vuelve al paso 2**

El “dataframe” implementado debe soportar columnas con los siguientes tipos de datos:

- Texto (cadenas de caracteres).
- Números (enteros y decimales).
- Fechas (en formato YYYY-MM-DD).

Para poder implementar el “dataframe” en C se van a utilizar las estructuras de datos que se describen a continuación:

```c
// Tipo enumerado para representar los diferentes tipos de datos en las columnas
typedef enum {
    TEXTO,
    NUMERICO,
    FECHA
} TipoDato;

// Estructura para representar una columna del dataframe
typedef struct {
    char nombre[30];       // Nombre de la columna
    TipoDato tipo;         // Tipo de datos de la columna (TEXTO, NUMERICO, FECHA)
    void *datos;           // Puntero genérico para almacenar los datos de la columna
    unsigned char *esNulo; // Array paralelo, indica valores nulos (1/0: nulo/noNulo)
    int numFilas;          // Número de filas en la columna
} Columna;

// Estructura para representar el dataframe como un conjunto de columnas
typedef struct {
    Columna *columnas;     // Array de columnas (con tipos de datos distintos)
    int numColumnas;       // Número de columnas en el dataframe
    int numFilas;          // Número de filas (igual para todas las columnas)
    int *indice;           // Array para ordenar las filas
} Dataframe;

// Alias para tipos FECHA: 'Fecha' alias de 'struct tm' (#include <time.h>)
typedef struct tm Fecha;
```

El tipo de datos enumerado (**enum**) se utiliza para definir los tipos de datos que soportará cada columna: **TEXTO**, **NUMERICO** (que puede incluir tanto enteros como decimales) y **FECHA** (en formato YYYY-MM-DD). Esto facilita la identificación y el tratamiento de cada tipo de columna durante las operaciones.

La estructura **Columna** representa una columna de datos del fichero CSV, contiene el **nombre** de dicha columna, el **tipo** de datos que almacena, y un puntero genérico **void *datos** que utilizará para almacenar el vector o array de datos. La ventaja de usar un puntero genérico (**void***) es que permite flexibilidad en los tipos de datos que se almacenan (ya sea un array de enteros, flotantes o cadenas de caracteres). El campo **esNulo** es un array paralelo que indica si un valor en la columna es nulo (1 para nulo, 0 para no nulo). Y **numFilas** especifica cuántas filas tiene la columna.

Dataframe es la estructura principal que representa un conjunto de datos. Contiene un array de estructuras **Columna**, lo que permite almacenar múltiples columnas con

diferentes tipos de datos. **numColumnas** y **numFilas** almacenan respectivamente el número de columnas y filas del conjunto de datos. Por último, la variable **indice** es un array de entrenos que se utilizará para realizar operaciones de ordenación de las filas del "dataframe".

En cuanto a la representación de fechas, una opción robusta en C es utilizar la estructura **struct tm** [↗] de la librería estándar **<time.h>** [↗], que proporciona un formato adecuado para manejar fechas con campos separados para el año, mes, día, etc. Esto facilitaría las operaciones de comparación y manipulación de fechas. El identificador **'Fecha'**, tal cual se ha definido, es un alias para la estructura '**struct tm**'.

Adicionalmente, el programa deberá poder trabajar con un número indefinido de dataframes, por lo que va a ser necesario disponer de una lista de dataframes, la cual se ha de implementar con las siguientes estructuras:

```
// Estructura para representar un nodo de la lista
typedef struct NodoLista{
    Dataframe *df;                // Puntero a un dataframe
    struct NodoLista *siguiente   // Puntero al siguiente nodo de la lista
} Nodo;

// Estructura para representar la lista de Dataframe's
typedef struct {
    int numDFs;       // Número de dataframes almacenados en la lista
    Nodo *primero;     // Puntero al primer Nodo de la lista
} Lista;
```

Inicialmente el programa no tendrá ningún dataframe, por lo que la lista estará vacía al arrancar. Para identificar un dataframe se usarán las palabras **df0**, **df1**, **df2**, ..., etc., es decir, los dataframes se irán numerando de cero (0) en adelante, según el orden en el que se encuentren en la lista.

El prompt del programa, el que debe mostrarse en la interfaz CLI, inicialmente tendrá esta forma:

[?]:>

Cuando haya un dataframe activo para trabajar con él, el prompt cambiará a una forma como la siguiente:

[df0: 100,5]:>

Este prompt indica que el dataframe activo es el **df0** (el primero de la lista) y que dicho dataframe tiene 100 filas y 5 columnas de datos. Cada vez que cambie el dataframe activo, deberá cambiar también el prompt de forma coherente, indicando su nombre y su número de filas y columnas.

COMANDOS

Junto al prompt debe aparecer un cursor para que el usuario del programa pueda introducir distintos comandos que el programa tiene que interpretar y ejecutar:

`quit`

Finaliza el programa, toda la memoria dinámica reservada debe liberarse, a continuación el programa terminará con un mensaje `EXIT PROGRAM` en verde, tras lo cual volverá a aparecer en la consola (o terminal) el `prompt de windows` en blanco.

`load <nombre_fichero>`

Carga un fichero de datos CSV (se mostrará un `mensaje de error` en rojo si ocurre algún error al intentar cargar los datos). Si la carga es correcta, se creará un nuevo dataframe y se convierte automáticamente en el dataframe activo (cambiará el prompt). Se entiende que la primera fila del dataframe indicará los nombres de las columnas de datos y la segunda fila (la primera con datos), determinará de qué tipo es cada columna (**`TEXTO`**, **`NUMERICO`** o **`FECHA`**). El separador será siempre una coma (,). Si al leer los datos se detecta un valor nulo, vacío o de un tipo incorrecto (según su columna) se indicará poniendo el valor '1' en la posición correspondiente del vector "**`esNulo`**".

`meta`

Este comando no lleva parámetros, en caso de indicarse alguno se considerará comando incorrecto. El comando muestra por pantalla los metadatos generales del dataframe activo, si no hay un dataframe activo también se considerará error. El comando mostrará por pantalla `en color verde` el nombre de cada columna de datos seguido de su tipo y el número de valores nulos que contiene (en una fila nueva cada columna). Se debe mostrar `en color rojo` cualquier mensaje de error, cuando sea el caso.

`view [n]`

Muestra por pantalla las '**`n`**' primeras filas del dataframe (o todas si hay menos de **`n`** filas). El parámetro '**`n`**' es opcional, en caso de indicarse, debe ser un número entero mayor que cero, si se omite el parámetro se muestran las 10 primeras filas del dataframe (o todas si hay menos de 10 filas). Para representar un valor nulo se imprimirá el código "**`#N/A`**" ('*Not Available*'). La salida por pantalla debe producirse en color verde, considerando el orden de las filas del dataframe indicado por el array "**`indice`**" de la estructura "**`Dataframe`**". Si corresponde indicar algún mensaje de error, este se indicará en color rojo.

`sort <nombre_columna> [asc/des]`

Se ordenan las filas del dataframe activo según la columna indicada, la ordenación se realizará utilizando el array "**`indice`**" de la estructura "**`Dataframe`**". Opcionalmente, se

puede indicar con el parámetro “**asc**” o “**des**” que la ordenación sea ascendente o descendente respectivamente, si no se indica este parámetro la ordenación será ascendente por defecto. La operación no se podrá realizar si no hay un dataframe activo o si el nombre de la columna indicado no existe en el dataframe activo, tampoco si hay algún parámetro mal escrito o hay más o menos parámetros de los necesarios, en todos esos casos deberá indicar un mensaje de error en rojo.

save <nombre_fichero>

Se guarda el dataframe actual en un fichero CSV cuyo nombre ha de ser el que se indica como parámetro (será error si no hay un dataframe activo). La primera línea del fichero CSV debe contener los nombres de las columnas, y se utilizará la coma (,) como separador de valores.

filter <nombre_columna> eq/neq/gt/lt <valor>

Esta operación consiste en aplicar un filtrado al dataframe actual, el filtrado se realizará en función de alguna de las columnas del dataframe, indicada con el parámetro “**nombre_columna**”, y el dato “**valor**” que se proporcione, aplicando uno de los cuatro operadores disponibles cuyo significado se explica a continuación:

- **eq**: “EQUAL” o “igual a”
- **neq**: “NOT EQUAL” o “distinto de”
- **gt**: “GREATER THAN” o “mayor que”
- **lt**: “LESS THAN” o “menor que”

La columna indicada y el valor empleado para filtrar deben ser del mismo tipo. Todas las filas que cumplan la condición indicada se conservarán y las que no cumplan la condición se borrarán del dataframe, el filtrado debe respetar cualquier ordenación que se hubiera realizado anteriormente. MUY IMPORTANTE, se debe hacer una buena gestión de la memoria dinámica.

delnull <nombre_columna>

Elimina del dataframe todas las filas que contengan un valor nulo en la columna indicada, al terminar se muestra por pantalla en verde un mensaje que indique el número de filas que se han borrado. El borrado debe respetar cualquier ordenación que se hubiera realizado anteriormente. MUY IMPORTANTE, se debe hacer una buena gestión de la memoria dinámica.

delcolum <nombre_columna>

Elimina del dataframe toda la columna indicada por completo. No se muestra ningún mensaje al hacer la operación, salvo que proceda algún mensaje de error. MUY IMPORTANTE, se debe hacer una buena gestión de la memoria dinámica.

quarter <nombre_columna> <nombre_nueva_columna>

Este comando crea una nueva columna de datos en el dataframe con valores de tipo **TEXTO**. La primera columna indicada debe existir en el dataframe actual y tiene que ser de tipo **FECHA**. La segunda columna indicada ("*nombre_nueva_columna*") es el nombre de la nueva columna que se ha de crear, no debe existir otra columna en el dataframe actual que se llame igual, esta nueva columna debe quedar a la derecha de la columna "*nombre_columna*". Los valores que debe contener la nueva columna son **Q1**, **Q2**, **Q3**, **Q4** o **#N/A**, dependiendo de si la fecha correspondiente pertenece al trimestre 1, 2, 3 o 4 del año o de si se trata de un valor nulo. MUY IMPORTANTE, se debe hacer una buena gestión de la memoria dinámica.

df0 / df1 / df2 / ...

Como se ha mencionado, los dataframes que se van almacenando en memoria se denominan '**df0**', '**df1**', '**df2**', etc. Cuando el comando sea el nombre de un dataframe, si existe, este se convertirá en el nuevo dataframe activo (modifica el prompt). Si el dataframe no existe se indica con un mensaje de error y continúa activo el mismo dataframe que había (o ninguno si no había ninguno activo). El nombre del dataframe tampoco debe ir acompañado de otros parámetros, eso será también un error.

REQUISITOS GENERALES

En el manejo del programa hay situaciones en las que un comando será incorrecto, tendrá parámetros incorrectos, o no se podrá ejecutar porque no se den ciertas condiciones. Cuando se dé alguna de estas circunstancias el programa deberá responder con un único mensaje de **error en color rojo**.

Cuando un comando se ejecute correctamente, solo si en la descripción del mismo se indica que debe mostrar alguna información por pantalla, esta **información se mostrará en color verde**. Si un comando se ejecuta correctamente pero en su descripción no se indica explícitamente que deba mostrar ninguna información, en ese caso, tras ejecutar el comando, simplemente se vuelve a imprimir el prompt para que el usuario pueda introducir un nuevo comando.

En ningún caso se mostrarán mensajes del tipo "*Pulse una tecla (o INTRO) para continuar*" o "*El comando se ha ejecutado correctamente*" ni ningún otro tipo de mensaje o información que no esté explícitamente indicado en este enunciado y, específicamente, en la descripción de los comandos que debe ejecutar el programa.

El prompt y el comando escrito por el usuario siempre se imprimen en color blanco.

IMPLEMENTACIÓN EN C

La práctica debe estar constituida por 3 ficheros:

- main.c ⇒ contendrá la función “main()” y, si el alumno lo considera oportuno, algunas definiciones u otras funciones secundarias.
- lib.h ⇒ fichero cabecera con las declaraciones de tipos y prototipos de funciones principales de la práctica. Todo el código que se ha escrito en este enunciado (páginas 2 y 3) debe incluirse en este fichero.
- lib.c ⇒ contendrá las definiciones de todas las funciones cuyos prototipos se han declarado en “lib.h”.

El código deberá poder compilar con el comando de consola/terminal:

gcc main.c lib.c (gcc main.c lib.c -o programa.exe)

Se recomienda implementar diversas funciones para toda la gestión de memoria necesaria para el desarrollo del programa, por ejemplo:

- Crear una columna
- Eliminar una columna
- Crear dataframe
- Eliminar dataframe
- Buscar columna en el dataframe
- Añadir dataframe a la lista
- Eliminar dataframe de la lista
- Eliminar toda la lista de dataframes
- Buscar dataframe en la lista
- Leer CSV desde fichero
- Guardar CSV en un fichero
- etc…

Recuérdese que se penalizará una mala gestión de memoria dinámica.

Ampliación 1
Examen Octubre/Noviembre 2024

 # En todos los ejercicios, cuando corresponda mostrar una salida por pantalla, se hará en color verde, salvo que sea un mensaje de error, que se hará en color rojo.

Avisos:

- La duración del examen es de 3 horas
- Si la práctica contiene algún error, este podría restar a la nota final del examen
- **MUY IMPORTANTE:** MOSTRAR POR PANTALLA AL PRINCIPIO DEL PROGRAMA QUE EJERCICIOS ESTÁN HECHOS Y CUÁLES NO **(no hacerlo restará 1 punto)**
    - `- Ejercicio 1: HECHO / SIN HACER`
    - `- Ejercicio 2: . . .`

Ampliación 2
Examen Enero 2025

 # En todos los ejercicios, cuando corresponda mostrar una salida por pantalla, se hará en color verde, salvo que sea un mensaje de error, que se hará en color rojo.

Avisos:

- La duración del examen es de 3 horas
- Si la práctica contiene algún error, este podría restar a la nota final del examen
- **MUY IMPORTANTE:** MOSTRAR POR PANTALLA AL PRINCIPIO DEL PROGRAMA QUE EJERCICIOS ESTÁN HECHOS Y CUÁLES NO **(no hacerlo restará 1 punto)**
    - `- Ejercicio 1: HECHO / SIN HACER`
    - `- Ejercicio 2: . . .`

Ampliación 3
Examen Junio/Julio 2025

---.

 # En todos los ejercicios, cuando corresponda mostrar una salida por pantalla, se hará en color verde, salvo que sea un mensaje de error, que se hará en color rojo.

Avisos:

- La duración del examen es de 3 horas
- Si la práctica contiene algún error, este podría restar a la nota final del examen
- **MUY IMPORTANTE:** MOSTRAR POR PANTALLA AL PRINCIPIO DEL PROGRAMA QUE EJERCICIOS ESTÁN HECHOS Y CUÁLES NO **(no hacerlo restará 1 punto)**
    - `- Ejercicio 1: HECHO / SIN HACER`
    - `- Ejercicio 2: . . .`