UNIVERSIDAD TECNICA
FEDERICO SANTA MARIA

ELO 320
Estructura de Datos y Algoritmos
Tarea #01
Mayo 30, 2021

DEPARTAMENTO DE ELECTRONICA

Semestre 2021 - 1

Profesor P1: Fernando Auat

Profesor P2: Ioannis Vourkas

Contenidos principales

- Llamado a funciones y uso de archivos de texto de I/O
- Uso de punteros y manejo de memoria dinámica
- Estructuras de datos: Lista enlazada, STACK, y Árbol Binario de Búsqueda (ABB)

Objetivos

- Familiarizarse con el desarrollo modular/incremental de código C
- Practicar con las estructuras de datos: Lista enlazada, STACK, y Árbol Binario de Búsqueda (ABB)

IMPORTANTE: **Las tareas son individuales**. Cualquier acción que pueda beneficiar de forma injusta la calificación de su tarea está prohibida, incluyendo la presentación de cualquier componente que no es de su autoría, o la facilitación de esto para otros. Es aceptable *discutir -en líneas generales- los métodos y resultados con sus compañeros*, pero **se prohíbe compartir soluciones de código. Utilizar código de internet que no es de su autoría, también es considerado plagio, a menos que se indique la fuente**. **Presten atención a las instrucciones de entrega**, que pueden incluir políticas de nombre de archivos y aspectos que se considerarán en la evaluación.

Descripción del trabajo a realizar

Parte A: *Preparación de datos de Entrada* (25 Pts)

Se entrega una lista de 25 alumnos (todos famosos) de un curso de EDA hipotético según su nota en el primer certamen. Los datos se encuentran en el archivo `notas-EDA-C1.txt`. En dicho archivo, cada línea tiene un solo string correspondiente a un alumno, conformado por 3 campos separados por una coma ',' de la siguiente forma:

```
Nombre,Apellido,nota
```

Su programa debe leer la información almacenada en el archivo `notas-EDA-C1.txt`, cuyo nombre se pasa como parámetro vía la línea de comandos. Para este propósito, se recomienda utilizar la función:

```
char * strtok(char * str, const char * delimiters)
```

(disponible en la biblioteca `string.h`) de manera repetitiva, para facilitar la identificación de los tres campos en cada línea del archivo.

Luego su programa debe encriptar el nombre y apellido de cada alumno, aplicando un algoritmo trivial de criptografía.

La encriptación se realizará ejecutando la función `encrypt(char* str, char table[ROWS][COLS])` que como segunda argumento recibirá el arreglo `char encryptionTable[2][26]` que en la primera fila tendrá todas las letras del alfabeto A … Z en orden, y en la segunda fila tendrá las letras desordenadas. Un ejemplo se muestra en la **Figura 1**.

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
F	Q	G	A	P	C	O	B	S	R	D	W	N	E	K	V	Z	L	J	U	X	I	Y	T	M	H

Fig. 1 Ejemplo de la tabla de encriptación

➔ Para llenar la segunda fila del arreglo, **usarán la función** `initEncryptionTable(char table[ROWS][COLS])` que, primero copia el contenido de la primera fila a la segunda, y posteriormente, para cada posición de la segunda fila, genera un número aleatorio de 0 a 25 e **intercambia** la letra correspondiente con la letra en tal **posición. Pero ¡OJO!, que tal intercambio de letras no debería nunca dejar la misma letra en la misma posición en ambas filas del arreglo.**

Para realizar el trabajo de **desencriptación**, su programa debe tener la función `decrypt(char* str, char table[ROWS][COLS])` que revelará el string original a partir de un string encriptado.

TIP: Cada vez que quieren **encriptar** una letra dentro de un string, deben recorrer la primera fila para encontrarla, y reemplazarla con la letra que se encuentra en la misma posición de la segunda fila, mientras que, para **desencriptar**, deben buscar en la segunda fila y reemplazar la letra con la que se encuentra en la misma posición de la primera fila.

A partir de esto, su programa debe **imprimir por pantalla** la información leída del archivo, pero con los datos de todos los alumnos encriptados, mostrando en cada línea lo siguiente:

`Nombre_encriptado,Apellido_encriptado,nota`

Parte B: *Uso de STACK y Listas Enlazadas* (40 Pts)

A medida que se vayan leyendo los datos del archivo, crearán un nuevo nodo para cada estudiante según la siguiente estructura, donde `nombre` y `apellido` serán encriptados:

```
#define NLEN 30

typedef struct node {
  char nombre[NLEN];
  char apellido[NLEN];
  int nota;
  struct node* previous;      //equivalente a left  en ABB
  struct node* next;          //equivalente a right en ABB
}Student;
```

Cada nodo será agregado a un **STACK** implementado con una lista doblemente enlazada. La interfaz para el manejo del STACK debe hacerse en los archivos **`myStack.h`** y **`myStack.c`**, que deben tener una estructura struct con las variables necesarias para administrar el Stack, e incluir como mínimo las funciones que se presentan (en forma genérica) a continuación:

```
isFull ()          // indica si el stack está lleno
isEmpty()          // indica si el stack está vacío
getNewStack ()     // genera un nuevo stack
stackDelete()      // elimina el stack y libera la memoria reservada
push()             // agrega un elemento al stack
pop()              // quita el primero elemento del stack
```

Al completar esta parte y tras formar el STACK con todos los alumnos, su programa debe imprimir por pantalla los nombres y apellidos **desencriptados** de los estudiantes que han aprobado la asignatura, y destacar el que sacó la mejor nota.

Parte C: *Uso de Árbol Binario de Búsqueda* (35 Pts)

En esta última parte, su programa debe quitar los nodos del STACK antes formado e **ingresarlos a un Árbol Binario de Búsqueda (ABB)** según el apellido de cada alumno. Asegúrense de que su ABB **acepte** notas repetidas.

En el contexto del ABB, en la estructura `struct node` puede considerar el campo `previous` como `left`, y el campo `next` como `right`. La interfaz para el manejo del ABB debe hacerse en los archivos **`myABB.h`** y **`myABB.c`**, que implementarán todas las funciones necesarias, incluidas las que son mencionadas a continuación (en forma genérica).

La inserción de nuevos nodos al ABB se realizará mediante la función `insertStudent()`. Una vez generado el ABB, su programa debe recorrerlo e imprimir por pantalla los nombres y apellidos **desencriptados** de los estudiantes que han aprobado la asignatura, **en orden alfabético**. Todas sus funciones y códigos en la parte C **NO pueden usar recursividad**.

Al completar este paso, su programa debe eliminar el ABB y también eliminar el STACK anteriormente creado, antes de terminar la ejecución.

Consideraciones y Formato de entrega

- Todas las funciones que NO forman parte de la interfaz del Stack y del ABB, serán definidas en el archivo `main.c` en la parte inferior, después de la función `main()`.
- Utilice **/*comentarios*/** para documentar lo que se hace en cada etapa de su código. **Se debe además usar el estilo de documentación Doxygen** utilizado en TODOS los ejemplos de código que encontrarán en AULA.
- El código deberá estar perfectamente *indentado* **con espacios**, no tabuladores
- Toda tarea debe ser correctamente **compilada** y ejecutada **en el servidor Aragorn (**tareas que **no compilan** se evaluarán cualitativamente y **tendrán como máximo nota 54)**.
- Para la entrega, suba **un archivo** comprimido (.zip, .rar, .tar.gz, etc.) que contenga su código (es decir, **SOLAMENTE** sus archivos **.c** y **.h**), más un archivo `README.txt`, donde especificará sus datos personales (nombre, apellido, ROL) y, de ser necesario, explicará cualquier particularidad que pueda tener su programa, como por ejemplo cómo debe ser compilado y ejecutado.

¿Cómo ENTREGAR?

El entregable debe ser subido a la plataforma **AULA**, **hasta las 23:50 del viernes 25 de junio, 2021**.