MEMORIA

P1 - AUTLEN

Diseño.-

El diseño se basa en las siguientes estructuras de datos: **estado, conjunto_simbolos, transición y AFND.**
En nuestra implementación un **estado** viene determinado por un identificador, un nombre y un tipo (indica si es inicial, final, normal o inicial final). **Conjunto_simbolos** es un a estructura de datos que nos vale para instanciar tanto cadenas de entrada al autómata como alfabetos (ya que en un array de C no podemos distinguir si importa el orden o no). Las cadenas lo único que se diferencian de los alfabetos es que tienen el nombre de "cadena" por defecto.

Transición lo hemos implementado como un conjunto de tablas, una por cada símbolo del alfabeto. Por su parte, cada tabla tiene dimensiones num_estados x num_estados, en las que cada celda es un 1 si hay transición entre esos dos estados o 0 en caso contrario.

En AFND nos hemos limitado a seguir la interfaz de funciones proporcionadas en el enunciado.

Algoritmo:

Condición de parada:
El algoritmo de procesamiento de la entrada tiene que estar activo hasta que se han procesado toda la cadena o si nos quedamos sin estados que analizar.
Descripción de iteración:
Por cada símbolo leído de la cadena, el algoritmo lee de la lista de los estados actuales y comprobado si cada uno de ellos tiene una transición del símbolo leído (en nuestro caso buscar en la tabla del símbolo) con cualquier otra estado del afnd y añadir este último a la nueva lista de actuales para la siguiente iteración.

Pruebas.-

En cuanto a las pruebas realizadas con el fin de comprobar el correcto funcionamiento de nuestro código, hemos desarrollado 4 módulos de test. Cada uno contiene un autómata con ciertas propiedades y se le pasan distintas cadenas de entrada. Al ejecutarlos uno a uno se puede comprobar que obtenemos la salida esperada por stdout. Todos los test han sido basados en autómatas propuestos en clase de teoría.

- Test1: el del enunciado
- Test2: ejemplo simple de clase
- Test3: presenta no determinismo
- Test4: su estado final coincide con el inicial

Herramientas.-

Para el correcto desarrollo de esta práctica hemos empleado diversas herramientas. En primer lugar codificamos mediante editores de texto habituales como **VSCode** o **Sublime** y el trabajo de compilado y ejecución ha sido automatizado por medio de un makefile. Para debuggear en ciertos momentos de necesidad hacemos uso de **gdb**.
En cuanto al manejo de memoria, nos aseguramos desde el primer momento de evitar fugas o accesos no permitidos mediante la herramienta **Memcheck** de Valgrind. Por último cabe destacar que para el control de versiones contamos con **Github** (De acuerdo con las normas de la escuela, tenemos el repositorio en modo privado), el cual hemos usado en todo momento para poder trabajar por separado.