**Proyecto de Lenguajes y Autómatas Miércoles 16-18**
**Guión análisis morfológico (tam1)**

**Objetivos**
Los objetivos de esta sesión son los siguientes:

- Familiarizarse con flex
- Estar preparados para completar el analizador morfológico y el programa principal que se pide en esta práctica.

**Material de apoyo**
Utilizaremos el **siguiente material**

- Gramática de ALFA.(está en Moodle, en la arte común)
- Presentación sobre flex (está en Moodle, en la arte común)
- Enunciado de la práctica (está en Moodle, en la parte común)
- Este guión (está en Moodle en nuestra sección)

*INTRODUCCIÓN A FLEX*
En grupos buscad en la documentación (consultando con el profesor) las siguientes cuestiones ya que

- ¿cuál es la entrada de flex?
- ¿qué hace flex?
- ¿cuál es el formato de los ficheros de especificación de flex?
- ¿qué es yyin?
- ¿qué es yylength?
- ¿qué es yylex?
- ¿qué es yyout?
- ¿qué es yywrap?
- ¿Cómo es la "lógica" de funcionamiento del programa producido por flex?

Se evaluará de manera individual

*TRABAJO PRÁCTICO CON FLEX*
Completa los siguientes apartados y sube a la entrega de Moodle un zip con los ficheros que hayas necesitado:

Los grupos de prácticas harán los siguientes ejercicios

1. Escribir un fichero de especificación flex (ej_flex_1.l) que contenga los patrones necesarios para identificar las siguientes palabras reservadas de alfa
    - main
    - array
    - int
    - if
    - else

- while

Debe crear un fichero .h en el que se defina un valor numérico para cada token que será el utilizado para comunicarse el .l con el programa principal

El fichero de especificación debe

- Utilizar en las acciones de cada patrón una sentencia C que realice un return del valor numérico asociado con cada token

El programa principal debe cumplir los siguientes requisitos

- El fichero que contiene el programa principal debe llamarse ej_flex_1.c
- Abrir como fichero de entrada aquel cuyo nombre se proporciona como primer argumento al invocar el programa (y ajustar las variables flex relacionadas con el fichero de entrada, si es necesario)
- Abrir como fichero de salida aquel cuyo nombre se proporciona como segundo argumento al invocar el programa (y ajustar las variables flex relacionadas con el fichero de salida, si es necesario)
- Escribir para cada token reconocido el siguiente mensaje (observe que para ello será necesario que ajuste sus programas C para que se acceda a la cadena en la que flex guarda el fragmento de entrada que corresponde con el último patrón reconocido)

RECONOCIDO <NUMERO DEL TOKEN>: <CADENA QUE CORRESPONDE CON ESE TOKEN>