# Teoría de la Computación

## Grado en Ingeniería Informática

- Prácticas de Laboratorio -

Profesor: *Gregorio de Miguel Casado*[*]

email: gmiguel@unizar.es

Dpto. de Informática e Ingeniería de Sistemas

Escuela de Ingeniería y Arquitectura

Universidad de Zaragoza

http://webdiis.unizar.es/asignaturas/TC/

Curso 2011-2012

[*]Material elaborado a partir de los guiones de prácticas creados y mantenidos por los profesores Pedro Álvarez, Rubén Béjar y Jorge Júlvez para la asignatura *Lenguajes Gramáticas y Autómatas* de la titulación *Ingeniería de Informática* (122) del plan de estudios BOE 1-2-1995.

# Práctica 1
# Introducción al manejo de *Flex*

## Tareas

1. Aprende a acceder a la página web de la asignatura, al entorno BlackBoard Learn y a trabajar con tu cuenta en Hendrix.
2. Descarga el documento `Intro_Flex_Bison.pdf` elaborado por el profesor Rubén Béjar Hernández y lee detenidamente el capítulo *'Introducción a Flex'*.
3. Lee la introducción de esta práctica y realiza los ejercicios 1 a 5 propuestos.
4. Elabora una memoria con los siguientes contenidos:
   - **Portada:** Número de Práctica, Grupo y Autor/Autores. Ejemplo:

     **Grupo COM1-M**
     **– Práctica 1 –**
     **Autores:** Al Anturing & Alon Zochurch

   - **Una sección para cada ejercicio resuelto.** Explica los aspectos fundamentales de la solución que propones y adjunta el código fuente. Ejemplo:

     **Ejercicio 1:**

     **1. Resumen** .....

     **2. Código fuente** .....

   - LA ENTREGA:
     - ¿QUÉ? Fichero en formato ***PDF***. Ejemplo:

       **COM1-M_P1_AlAnturing_AlonZochurch.pdf**

     - ¿CÓMO? Usando la plataforma **BlackBoard Learn**.
     - ¿CUÁNDO? Hasta las **23:59:59 del día anterior a la tercera sesión de prácticas**.

**Nota: El incumplimiento de las normas de entrega se reflejará en la calificación de la memoria.**

## Introducción

El objetivo principal de esta primera practica de la asignatura es familiarizarse con la herramienta de creación de analizadores léxicos *Flex* y aprender aspectos básicos sobre teoría de lenguajes y expresiones regulares. Para ello, se propone la creación con dicha herramienta de una serie de pequeños procesadores de texto

Las prácticas de TC se realizan en **hendrix-ssh.cps.unizar.es**. Para compilar los programas, los ejecutables correspondientes a *Flex* y *Bison* están en los directorios en */opt/csw/bin/flex* y */opt/csw/bin/bison*, respectivamente. Ejemplo:

```
flex nombre_fichero_fuente
gcc lex.yy.c -lfl -o nombre_ejecutable
```

El alfabeto $\Sigma$ que maneja *Flex* está formado por los caracteres manejables por el sistema (p. ej. todos los símbolos del código ASCII). En las prácticas se trabaja con letras (distinguiendo entre mayúsculas y minúsculas), números y signos de puntuación. El cuadro 1.1 muestra algunas correspondencias entre la notación empleada en clase de teoría y la que se utiliza en *Flex* para trabajar con Expresiones Regulares (ERs).

| **Operación** | **Notación Teoría** | **Notación Flex** |
|---|---|---|
| Concatena ERs | $\cdot$ ó {vacío} | {vacío} (yuxtaposición) |
| Unión de ERs | $+$ | $\vert$ |
| Cerradura de Kleene | $^*$ | $*$ |
| Cerradura Positiva | $^+$ | $+$ |
| Paréntesis | ( ) | ( ) |
| Conjunto a,b,c,d,0,1,2 | $\{a, b, c, d, 0, 1, 2\}$ | $[abcd012]$ ó $[a - d0 - 2]$ |
| {vacío} ó a | $\epsilon + a$ | $a?$ |
| Repetir 0 veces | $\epsilon$ | Sin equivalencia (ver $*$ ó ?) |

Cuadro 1.1: Correspondencia entre notaciones

## Ejercicio 1

Escribe un programa con *Flex* que elimine los *caracteres blancos* al final de cada línea del texto de entrada y reduzca a un solo *carácter blanco* cualquier grupo de éstos en el interior de cada línea. Se entiende por *carácter blanco* el espacio en blanco y el carácter de tabulación. Ejemplo:

**Entrada:**
```
Este es    un   texto <EOL>
con      varios blancos <EOL>
<EOF>
```

**Salida:**
```
Este es un texto <EOL>
con varios blancos <EOL>
<EOF>
```

## Ejercicio 2

Elabora un programa en *Flex* que copie el archivo de entrada en uno de salida, sustituyendo todo numero entero positivo múltiplo de 7 por el carácter '$*$'.

Este ejercicio tiene al menos dos enfoques:

- Construir una ER que denote el lenguaje de las palabras sobre el conjunto de símbolos 0,1,2,3,4,5,6,7,8,9 y que en decimal sean múltiplo de 7 para luego buscarla en *Flex*.

  ¿Se puede construir una ER que reconozca los números decimales múltiplos de 7? ¿Cómo sería la expresión?
- Construir una ER para leer números en decimal, y después en la acción asociada en *Flex* comprobar si éstos son múltiplos de 7.

## Ejercicio 3

Construye un programa en *Flex* que cuente el número de palabras del texto de entrada que contengan alguna vez el carácter 'w'. ¿Cómo sería la ER? ¿Y cómo sería la ER si queremos identificar las palabras que contengan un solo carácter 'w'?

## Ejercicio 4

Escribe un programa en *Flex* que para una secuencia de entrada dada, realice lo siguiente para cada uno de estos casos:

- Informar del número de caracteres, palabras y líneas de texto.
- Reconocer los números enteros, calcular e informar de su media.
- Reconocer los números reales e indicar cuántos de ellos están comprendidos en el rango entre cero y uno.

**Nota:** La principal dificultad de este ejercicio está en saber tratar los números como palabras compuestas por caracteres numéricos. Además, hay que tener en cuenta que un carácter del fichero de entrada en *Flex* sólo se puede emparejar con un solo patrón (ER) como máximo.

## Ejercicio 5

Elabora un programa con *Flex* que, a partir de un fichero de texto que se pasa a la entrada, verifique que siempre que se abre un paréntesis, se cierre. Si detecta algún error, se informará de la línea donde se produce dicho error. La dificultad de este ejercicio radica en saber responder a las siguientes cuestiones:

- ¿Es regular el lenguaje de las palabras sobre el alfabeto $\{(,)\}$ que representa todos los pares de paréntesis correctamente anidados? ¿Puedes demostrarlo?
- Si no fuera regular, ¿Se puede hacer algo?