# Análisis Léxico.
# Un Scanner para MiniLan (II)

Autómatas y Matemáticas Discretas
Escuela de Ingeniería Informática
Universidad de Oviedo
Curso 2017-2018

En esta sesión se completará el conjunto de macros y expresiones regulares de la especificación léxica del lenguaje MiniLan, completando el trabajo de las sesiones previas. Para iniciar esta sesión, cámbiate de nuevo al directorio ∼/`practicasAMD/LexicalAnalysis/ScannerMiniLan` y realiza los siguientes ejercicios.

## 1. Ejercicio: reconocer nuevos tokens

Modifica tu fichero de especificación para reconocer los siguientes tokens:

1. **Nuevos operadores**: añade las reglas para `<`, `>` and `==`. Los mensajes a imprimir en cada caso deben ser:
   `SCANNER:: found Operator` seguido por `LT`, `GT` o `EQ`, respectivamente.
2. **Comentarios**
   - Escribe una macro denominada `LETTER` que case con una letra, bien sea minúscula o mayúscula.
   - Escribe una macro denominada `BLANK` que case con un espacio en blanco o con un tabulador.
   - Usando las dos macros anteriores, escribe la macro `COMMENT` que case con las líneas de comentario definidas para `MiniLan`.
   - Añade una regla que expanda la macro `COMMENT`. El mensaje de salida debe ser:
     `SCANNER:: comment line <texto del comentario>`.
3. **Números reales**
   - Añade una macro denominada `REAL`, su misión es definir el patrón para los números reales en MiniLan.
   - Añade la regla correspondiente a los números reales, de manera que expanda la macro `REAL`. El mensaje de salida debe ser `SCANNER:: found REAL <XXXX>`, donde XXXX se corresponde con el valor numérico detectado.
4. Vamos a comprobar que todo va bien, genera el Scanner y comprueba su funcionalidad con el archivo de test `testML2`, cuyo contenido se muestra a continuación.

```
print begin end
+ - *// () <==>
321 4.35 .345 43.
. // dot is not a real number
```

Dispones de una copia del mismo en el directorio: `/var/asignaturas/AMD/practicas/test_files/`
La salida por pantalla sería la siguiente:

```
SCANNER:: found Reserved Word PRINT
SCANNER:: found Reserved Word BEGIN
SCANNER:: found Reserved Word END
SCANNER:: found Operator ADD
SCANNER:: found Operator MINUS
SCANNER:: found Operator MULT
SCANNER:: comment line <// >
SCANNER:: found symbol RP
SCANNER:: found symbol LP
SCANNER:: found Operator LT
SCANNER:: found Operator EQ
SCANNER:: found Operator GT
SCANNER:: found NUMBER <321>
SCANNER:: found REAL <4.35>
SCANNER:: found REAL <.345>
SCANNER:: found REAL <43.>
SCANNER:: Unmatched input .
SCANNER:: comment line <// dot is not a real number>
```

## 2. Ejercicio: completando MiniLan.lex

Para finalizar nuestro analizador léxico para MiniLan, añade el codigo necesario para reconocer los siguientes tokens.

- **Operador de asignación**(=): el mensaje a mostrar por pantalla será:
  `SCANNER:: found Operator SET`:
- **Identificadores**: utilizando las macros LETTER y DIGIT definir otra macro IDENT para los identificadores de MiniLan. Añade una regla para esta macro cuya salida sea:
  `SCANNER:: found IDENT <ident>`, donde *ident* es el identificador leído

## 3. Prueba final

Para probar el funcionamiento de nuestro analizador léxico de MiniLan utiliza el fichero de test `testML3` y examina si tu salida coincide con la que está a continuación. Dispones de una copia del mismo en el directorio: `/var/asignaturas/AMD/practicas/test_files/`

Fichero *testML3*

```
print begin end
+ - *// () <==>
321 4.35 .345 43.
. // dot is not a real number

Suma = 3. + 1 / .8 * tend
```

La salida por pantalla sería la siguiente:

```
SCANNER:: found Reserved Word PRINT
SCANNER:: found Reserved Word BEGIN
SCANNER:: found Reserved Word END
SCANNER:: found Operator ADD
SCANNER:: found Operator MINUS
SCANNER:: found Operator MULT
SCANNER:: comment line <// >
SCANNER:: found symbol RP
SCANNER:: found symbol LP
SCANNER:: found Operator LT
SCANNER:: found Operator EQ
SCANNER:: found Operator GT
SCANNER:: found NUMBER <321>
SCANNER:: found REAL <4.35>
SCANNER:: found REAL <.345>
SCANNER:: found REAL <43.>
SCANNER:: Unmatched input .
SCANNER:: comment line <// dot is not a real number>
SCANNER:: found IDENT <Suma>
SCANNER:: found Operator SET
SCANNER:: found REAL <3.>
SCANNER:: found Operator ADD
SCANNER:: found NUMBER <1>
SCANNER:: found Operator DIV
SCANNER:: found REAL <.8>
SCANNER:: found Operator MULT
SCANNER:: found IDENT <tend>
```