

# Desarrollo de analizadores léxicos

## Grupo 18

### Alumnos:

Jorge Torres Ruiz  
Alvaro Montes Anacona  
Daniel López Escobar  
Ignacio García Fernández

### ***1.1 Enumeración de las clases léxicas de Tiny(0).***

#### Clases auxiliares

**letra:** Clase que comprende las letras desde la a a la z tanto en mayúsculas como minúsculas

**subrayado:** Clase que comprende el símbolo de subrayado

**dígito\_positivo:** Clase que comprende los números del 1 al 9

**dígito:** Representa los dígitos positivos añadiendo el 0

**parte\_entera:** Clase correspondiente a la parte entera de los números reales

**parte\_decimal:** Se corresponde con la parte decimal de los números reales

**parte\_exponencial:** Clase para la parte exponencial que pueda aparecer en los reales

#### Clases "principales"

**identificador:** Nombres que reciben las variables, Comienzan necesariamente por una letra o subrayado (\_), seguida de una secuencia de cero o más letras, dígitos, o subrayado (\_)

**literal\_entero:** representación de los números enteros, podrán comenzar con signo opcionalmente, seguido de una secuencia de al menos un dígito (no ceros no significativos)

**literal\_real:** representa un número real con parte entera y luego parte decimal o parte exponencial o ambas

**suma:** para representar el operador de la suma entre dos elementos

**resta:** para representar el operador de la resta entre dos elementos

**multiplicación:** representar el operador de multiplicación entre dos elementos

**división:** representar el operador de división entre dos elementos

**menor\_que:** clase léxica para representar el operando de comparación menor estricto  
**mayor\_que:** clase léxica para representar el operando de comparación mayor estricto  
**menor\_igual\_que:** clase léxica para representar el operando de comparación menor o igual  
**mayor\_igual\_que:** clase léxica para representar el operando de comparación mayor o igual  
**comparación:** representar la operación de comparación entre dos expresiones  
**distinto\_que:** representar la operación que determina si dos expresiones tienen distinto valor  
**asignación:** para representar la asignación de una expresión o literal a una variable  
**parAP:** representar el paréntesis de apertura  
**parCIERR:** representar el paréntesis de cierre  
**fin\_declaraciones:** para representar las dos ampersand de fin de la sección de declaraciones  
**inicio\_eval:** representar el símbolo que da comienzo a una instrucción de evaluación  
**punto\_y\_coma:** representación del símbolo que separa declaraciones y evaluaciones

**int:** representar dicha palabra reservada del lenguaje  
**real:** representar dicha palabra reservada del lenguaje  
**bool:** representar dicha palabra reservada del lenguaje  
**and:** representar dicha palabra reservada del lenguaje  
**or:** representar dicha palabra reservada del lenguaje  
**not:** representar dicha palabra reservada del lenguaje

**comentario:** representa un comentario del lenguaje (cadenas ignorables)

## **1.2 Especificación formal del léxico del lenguaje mediante definiciones regulares**

### **Definiciones Auxiliares**

**letra** = [a-z, A-Z, \_]

**dígito\_positivo**: [1-9]

**dígito**: {dígito\_positivo} | 0

**signo**: (\+, \-)

**parte\_entera**: {signo}? ( {dígito\_positivo} {dígito}\* | 0 )

**parte\_decimal**: ( {dígito}\* {dígito\_positivo} | 0 )

**parte\_exponencial**: (e | E) {parte\_entera}

### **Clases léxicas**

**identificador**: (letra) ( {letra} | {dígito})\*

**literal\_entero**: {parte\_entera}

**literal\_real**: {literal\_entero} ({parte\_exponencial} | \.{parte\_decimal} |  
 \.{parte\_decimal}{parte\_exponencial})

**suma**: \+

**resta**: \-

**multiplicación**: \\*

**división**: /

**menor\_que**: <

**mayor\_que**: >

**menor\_igual\_que**: <=

**mayor\_igual\_que**: >=

**comparación**: ==

**distinto\_que**: !=

**parAP**: \ (

**parCIERR**: \ )

**fin\_declaraciones**: &&

**inicio\_eval**: @

**punto\_y\_coma**: ;

**asignación**: =

**llaveAP**: \ {

**llaveCIERR**: \ }

### **Palabras reservadas**

**Int**: [i, l][n, N][t, T]

**Real**: [r, R][e, E][a, A][l, L]

**Bool**: [b, B][o, O][o, O][l, L]

**And:** [a, A][n, N][d, D]

**Or:** [o, O][r, R]

**Not:** [n, N][o, O][t, T]

**True:** [t, T][r, R][u, U][e, E]

**False:** [f, F][a, A][l, L][s, S][e, E]

**Cadenas ignorables**

**comentario** `##([^\n,EOF])*`


**separador** `[ ,\t,\r,\b,\n]`

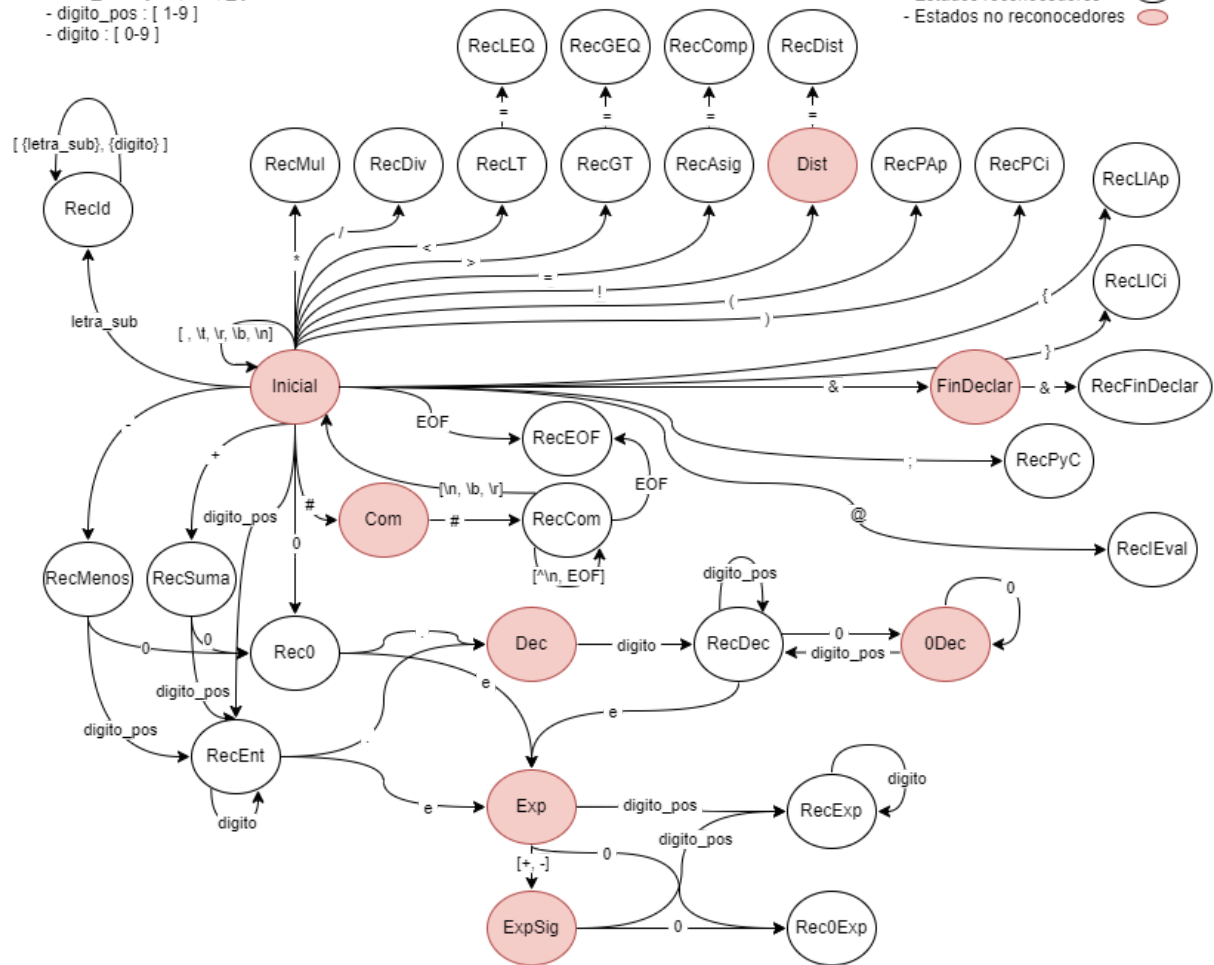
### Diagrama de transiciones

Clases Auxiliares:

- letra\_sub : [ a-z, A-Z, \_ ]
- digito\_pos : [ 1-9 ]
- digito : [ 0-9 ]

Leyenda:

- Estados reconocedores 
- Estados no reconocedores 



**TINY** (Las marcadas en azul son las que se añaden a tiny(0) para crear tiny)

### 1.3 Especificación formal del léxico del lenguaje mediante definiciones regulares

#### Clases auxiliares

**letra** = [a-z, A-Z, \_]  
**dígito\_positivo**: [1-9]  
**dígito**: {dígito\_positivo} | 0  
**parte\_entera**: ( {dígito\_positivo} {dígito}\* | 0 )  
**parte\_decimal**: ( {dígito} \* {dígito\_positivo} | 0 )  
**parte\_exponencial**: (e | E) {parte\_entera}  
**separador** [ , \t, \r, \b, \n]

#### Clases léxicas

**identificador**: ( letra ) ( {letra} | {dígito} )<sup>\*</sup>  
**literal\_entero**: [+, -] ? {parte\_entera}  
**literal\_real**: {literal\_entero} ({parte\_exponencial} | \.{parte\_decimal} |  
 \.{parte\_decimal}{parte\_exponencial})  
**cadena**: “ ( [^ ” ] )<sup>\*</sup> ”  
**suma**: \+  
**resta**: \-  
**multiplicación**: \\*  
**división**: /  
**porcentaje**: %  
**menor\_que**: <  
**mayor\_que**: >  
**menor\_igual\_que**: <=  
**mayor\_igual\_que**: >=  
**comparación**: ==  
**asignación**: =  
**distinto\_que**: !=  
**parAP**: \  
**parCIERR**: \)  
**fin\_declaraciones**: &&  
**inicio\_eval**: @  
**punto\_y\_coma**: ;  
**corchAP**: \[  
**corchCIERR**: \]  
**punto**: .

**exponente:** ^

**ampersand:** &

**llaveAP:** \{

**llaveCIERR:** \}

### Palabras reservadas

**Int:** [i, I][n, N][t, T]

**Real:** [r, R][e, E][a, A][l, L]

**Bool:** [b, B][o, O][o, O][l, L]

**And:** [a, A][n, N][d, D]

**Or:** [o, O][r, R]

**Not:** [n, N][o, O][t, T]

**String:** [s, S][t, T][r, R][i, I][n, N][g, G]

**Null:** [n, N][u, U][l, L][l, L]

**True:** [t, T][r, R][u, U][e, E]

**False:** [f, F][a, A][l, L][s, S][e, E]

**Proc:** [p, P][r, R][o, O][c, C]

**If:** [i, I][f, F]

**Else:** [e, E][l, L][s, S][e, E]

**While:** [w, W][h, H][i, I][l, L][e, E]

**Struct:** [s, S][t, T][r, R][u, U][c, C][t, T]

**New:** [n, N][e, E][w, W]

**Delete:** [d, D][e, E][l, L][e, E][t, T][e, E]

**Read:** [r, R][e, E][a, A][d, D]

**Write:** [w, W][r, R][i, I][t, T][e, E]

**NI:** [n, N][l, L]

**Type:** [t, T][y, Y][p, P][e, E]

**Call:** [c, C][a, A][l, L][l, L]

### Cadenas ignorables

**comentario** ##([^\n,EOF])\*