

---

# Procesadores de Lenguajes

---

Memoria de proyecto — Hito 2: Analizador Sintáctico

## GRUPO 14

RODRIGO SOUTO SANTOS  
LEONARDO PRADO DE SOUZA  
JUAN ANDRÉS HIBJAN CARDONA  
IZAN RODRIGO SANZ

*Grado en Ingeniería informática  
Facultad de Informática  
Universidad Complutense de Madrid*



# Índice general

<b>1. Tiny (0)</b>	<b>2</b>
1.1. Introducción . . . . .	2
1.2. Clases léxicas . . . . .	2
1.2.1. Palabras reservadas . . . . .	2
1.2.2. Literales . . . . .	2
1.2.3. Identificadores . . . . .	2
1.2.4. Símbolos de operación y puntuación . . . . .	3
1.3. Especificación formal del léxico . . . . .	3
1.3.1. Definiciones auxiliares. . . . .	3
1.3.2. Definiciones léxicas. . . . .	3
1.3.3. Definiciones de cadenas ignorables. . . . .	4
1.4. Diseño de un analizador léxico . . . . .	4
<b>2. Tiny</b>	<b>5</b>
2.1. Introducción . . . . .	5
2.2. Clases léxicas . . . . .	5
2.2.1. Palabras reservadas . . . . .	5
2.2.2. Literales . . . . .	6
2.2.3. Identificadores . . . . .	6
2.2.4. Símbolos de operación y puntuación . . . . .	6
2.3. Especificación formal del léxico . . . . .	7
2.3.1. Definiciones auxiliares. . . . .	7
2.3.2. Definiciones léxicas. . . . .	7
2.3.3. Definiciones de cadenas ignorables. . . . .	8
<b>3. Tiny (0)</b>	<b>9</b>
3.1. Especificación Sintáctica (Gramática) . . . . .	9
3.1.1. Declaraciones . . . . .	9
3.1.2. Tipos . . . . .	9
3.1.3. Instrucciones . . . . .	9
3.1.4. Expresiones . . . . .	9
3.1.5. Operadores . . . . .	10
3.2. Acondicionamiento . . . . .	10
3.2.1. Declaraciones . . . . .	10
3.2.2. Tipos . . . . .	10
3.2.3. Instrucciones . . . . .	10
3.2.4. Expresiones . . . . .	11
3.2.5. Operadores . . . . .	11
3.3. Directores . . . . .	11
3.3.1. Tabla de Reglas . . . . .	11
<b>4. Tiny</b>	<b>14</b>
4.1. Especificación Sintáctica (Gramática) . . . . .	14
4.1.1. Declaraciones . . . . .	14
4.1.2. Tipos . . . . .	14
4.1.3. Instrucciones . . . . .	14
4.1.4. Expresiones . . . . .	15
4.1.5. Operadores . . . . .	15
4.2. Acondicionamiento . . . . .	16
4.2.1. Declaraciones . . . . .	16
4.2.2. Tipos . . . . .	16
4.2.3. Instrucciones . . . . .	17
4.2.4. Expresiones . . . . .	17
4.2.5. Operadores . . . . .	18
<b>Índice de figuras</b>	<b>19</b>
<b>Índice de cuadros</b>	<b>20</b>

# 1 | Tiny (0)

---

## 1.1. Introducción

Para realizar este apartado nos hemos fijado en todas las funcionalidades que aparecen en el “Apendice A” del archivo “fase1.pdf”. En los siguientes apartados definimos todas las clases que hay, su correspondiente especificación y un diagrama de transiciones.

## 1.2. Clases léxicas

### 1.2.1. Palabras reservadas

Para poder analizar de manera correcta, será necesario establecer una clase léxica por cada palabra reservada. En el lenguaje de esta práctica, *Tiny (0)*, contamos con 8 palabras reservadas, 3 de ellas utilizadas para definir el tipo de las variables. Tendremos pues, una palabra para las variables de tipo booleano, otra para las de tipo entero y una última para las reales. Además de éstas tendremos 5 palabras utilizadas para las operaciones lógicas. Las palabras son las definidas a continuación, contando cada una con una clase léxica.

- *bool* → Variables booleanas.
- *int* → Variables enteras.
- *real* → Variables reales.
- *and* → Conjunción lógica.
- *or* → Disyunción lógica.
- *not* → Negación lógica.
- *true* → Valor booleano cierto.
- *false* → Valor booleano falso.

### 1.2.2. Literales

- **Literales enteros.** Opcionalmente empiezan con un signo más (+) o menos (-), y después debe aparecer una secuencia (que empieza por un número distinto de 0) de 1 o más dígitos. Su clase léxica será *literalEntero*.
- **Literales reales.** Empieza con una parte entera seguida bien de una parte decimal, bien de una exponencial o bien una parte decimal seguida de exponencial. La parte decimal comienza con el signo punto (.) seguido de una secuencia (que puede ser sólo un 0 o números que no acaben en 0) de 1 o más dígitos. La parte exponencial se indica con (e) o (E), seguida de una parte entera. Su clase léxica será *literalReal*.

### 1.2.3. Identificadores

Los identificadores nos sirven para poder ponerle un nombre a las variables. Éstos deben comenzar por un subrayado (\_) o una letra, seguida de una secuencia de 0 o más subrayados, dígitos o letras. Su clase léxica será *identificador*.

### 1.2.4. Símbolos de operación y puntuación

Cada uno de ellos tendrá su propia clase léxica. En el subconjunto del lenguaje en el que trabajamos, *Tiny (0)*, contamos con las siguientes clases:

- **Suma.** Se representa con el símbolo más (+). Su clase léxica será *suma*.
- **Resta.** Se representa con el símbolo menos (-). Su clase léxica será *resta*.
- **Multiplicación.** Se representa con el símbolo asterisco (\*). Su clase léxica será *mul*.
- **División.** Se representa con el símbolo barra (/). Su clase léxica será *div*.
- **Menor.** Se representa con el símbolo menor que (<). Su clase léxica será *menor*.
- **Mayor.** Se representa con el símbolo mayor que (>). Su clase léxica será *mayor*.
- **Igual.** Se representa con dos símbolos de igualdad seguidos (==). Su clase léxica será *igual*.
- **Menor o igual.** Se representa con el símbolo menor que seguido del símbolo de igualdad (<=). Su clase léxica será *menorIgual*.
- **Mayor o igual.** Se representa con el símbolo mayor que seguido del símbolo de igualdad (>=). Su clase léxica será *mayorIgual*.
- **Asignación.** Se representa con un símbolo de igualdad (=). Su clase léxica será *asig*.
- **Final.** Se representa con el símbolo ampersand dos veces consecutivas (&&). Su clase léxica será *finalAsig*.
- **Paréntesis de apertura.** Se representa con el símbolo del paréntesis de apertura ("(", sin comillas). Su clase léxica será *parenApert*.
- **Paréntesis de cierre.** Se representa con el símbolo del paréntesis de cierre (")", sin comillas). Su clase léxica será *parenCierre*.
- **Llave de apertura.** Se representa con el símbolo de la llave de apertura ("{"", sin comillas). Su clase léxica será *LlaveApert*.
- **Llave de cierre.** Se representa con el símbolo de la llave de cierre ("}", sin comillas). Su clase léxica será *LlaveCierre*.
- **Punto y coma.** Se representa con el símbolo punto y coma (;). Su clase léxica será *puntoComa*.
- **Arroba.** Se representa con el símbolo arroba (@). Su clase léxica será *arroba*.

## 1.3. Especificación formal del léxico

### 1.3.1. Definiciones auxiliares.

$letra \rightarrow [a - z, A - Z]$   
 $digitoPositivo \rightarrow [1 - 9]$   
 $digito \rightarrow digitoPositivo | 0$   
 $parteEntera \rightarrow [\backslash +, \backslash -]? (\{ digitoPositivo \} \{ digito \}^* | 0)$   
 $parteDecimal \rightarrow (\{ digito \}^* \{ digitoPositivo \} | 0)$   
 $parteExponencial \rightarrow (e | E) parteEntera$

### 1.3.2. Definiciones léxicas.

$bool \rightarrow (b | B) (o | O) (o | O) (l | L)$   
 $int \rightarrow (i | I) (n | N) (t | T)$   
 $real \rightarrow (r | R) (e | E) (a | A) (l | L)$   
 $and \rightarrow (a | A) (n | N) (d | D)$   
 $or \rightarrow (o | O) (r | R)$   
 $not \rightarrow (n | N) (o | O) (t | T)$   
 $true \rightarrow (t | T) (r | R) (u | U) (e | E)$

### 1.3.3. Definiciones de cadenas ignorables.

#### 1.4. Diseño de un analizador léxico

---

---

CAPÍTULO 1. TINY (0)

## 2 | Tiny

---

### 2.1. Introducción

Para realizar este apartado nos hemos basado en todas las funcionalidades que aparecen en el archivo “lenguaje.pdf” que se ha aportado en el campus. En los siguientes apartados definimos todas las clases que hay y su correspondiente especificación.

### 2.2. Clases léxicas

#### 2.2.1. Palabras reservadas

Para poder analizar de manera correcta, será necesario establecer una clase léxica por cada palabra reservada. En el lenguaje de esta práctica, *Tiny*, contamos con 3 palabras reservadas, utilizadas para definir el tipo de las variables. Tendremos pues, una palabra para las variables de tipo booleano, otra para las de tipo entero y una última para las reales. También contamos con 5 palabras reservadas para los operadores lógicos and, or, not, true y false, 1 palabra reservada para hacer referencia a la nada, 1 palabra reservada para referenciar una función, 3 palabras reservadas para control de flujo, 1 palabra reservada para la creación de un estructura, 1 palabra reservada para reserva de memoria, 1 palabra reservada para liberar la memoria, 1 palabra reservada para lectura, 1 palabra reservada para escritura, 1 palabra reservada para nueva línea, 1 palabra reservada para vínculos de los nombres de tipo y 1 palabra reservada para invocación a procedimiento. Las palabras son las definidas a continuación, contando cada con una clase léxica.

- *bool* → Variables booleanas.
- *int* → Variables enteras.
- *real* → Variables reales.
- *string* → Variables de cadena.
- *and* → Conjunción lógica.
- *or* → Disyunción lógica.
- *not* → Negación lógica.
- *true* → Valor booleano cierto.
- *false* → Valor booleano falso.
- *null* → Referencia a la nada.
- *proc* → Función.
- *if* → Condición.
- *else* → Condición alternativa.
- *while* → Bucle con condición.
- *struct* → Estructura.
- *new* → Reserva de memoria.
- *delete* → Liberación de memoria.
- *read* → Lectura.
- *write* → Escritura.
- *nl* → Nueva línea.
- *type* → Vinculo de tipo.

- *call* → Invocación procedimiento.

### 2.2.2. Literales

- **Literales enteros.** Opcionalmente empiezan con un signo más (+) o menos (-), y después debe aparecer una secuencia (que empieza por un número distinto de 0) de 1 o más dígitos. Su clase léxica será *literalEntero*.
- **Literales reales.** Empieza con una parte entera seguida de una parte decimal, exponencial o parte decimal seguida de exponencial. La parte decimal comienza con el signo punto (.) seguido de una secuencia (que puede ser sólo un 0 o números que no acaben en 0) de 1 o más dígitos. Por último, y también opcionalmente, puede aparecer una parte exponencial que se indica con (e) o (E), seguida de una parte entera con o sin parte decimal. Su clase léxica será *literalReal*.
- **Literales de cadena.** Secuencia de 0 o más caracteres distintos que están entre comillas dobles (“ ”). Los caracteres pueden incluir las siguientes secuencias de escape: retroceso (`\b`), retorno de carro (`\r`), tabulador (`\t`) y salto de línea (`\n`). Su clase léxica será *literalCadena*.

### 2.2.3. Identificadores

Los identificadores nos sirven para poder ponerle un nombre a las variables. Éstos deben comenzar por un subrayado (`_`) o una letra, seguida de una secuencia de 0 o más subrayados, dígitos o letras. Su clase léxica será *identificador*.

### 2.2.4. Símbolos de operación y puntuación

Cada uno de ellos tendrá su propia clase léxica y son las siguientes clases:

- **Suma.** Se representa con el símbolo más (+). Su clase léxica será *suma*.
- **Resta.** Se representa con el símbolo menos (-). Su clase léxica será *resta*.
- **Multiplicación.** Se representa con el símbolo asterisco (\*). Su clase léxica será *mul*.
- **División.** Se representa con el símbolo barra (/). Su clase léxica será *div*.
- **Menor.** Se representa con el símbolo menor que (<). Su clase léxica será *menor*.
- **Mayor.** Se representa con el símbolo mayor que (>). Su clase léxica será *mayor*.
- **Igual.** Se representa con dos símbolos de igualdad seguidos (==). Su clase léxica será *igual*.
- **Menor o igual.** Se representa con el símbolo menor que seguido del símbolo de igualdad (<=). Su clase léxica será *menorIgual*.
- **Mayor o igual.** Se representa con el símbolo mayor que seguido del símbolo de igualdad (>=). Su clase léxica será *mayorIgual*.
- **Asignación.** Se representa con un símbolo de igualdad (=). Su clase léxica será *asig*.
- **Final.** Se representa con el símbolo ampersand dos veces consecutivas (&&). Su clase léxica será *finalAsig*.
- **Paréntesis de apertura.** Se representa con el símbolo del paréntesis de apertura (“(”, sin comillas). Su clase léxica será *parenApert*.
- **Paréntesis de cierre.** Se representa con el símbolo del paréntesis de cierre (“)”, sin comillas). Su clase léxica será *parenCierre*.
- **Llave de apertura.** Se representa con el símbolo de la llave de apertura (“{”, sin comillas). Su clase léxica será *LlaveApert*.
- **Llave de cierre.** Se representa con el símbolo de la llave de cierre (“}”, sin comillas). Su clase léxica será *LlaveCierre*.
- **Punto y coma.** Se representa con el símbolo punto y coma (;). Su clase léxica será *puntoComa*.
- **Arroba.** Se representa con el símbolo arroba (@). Su clase léxica será *arroba*.

- **Módulo.** Se representa con el símbolo barra (%). Su clase léxica será *mod*.
- **Coma.** Se representa con el símbolo coma (,). Su clase léxica será *coma*.
- **Indirección.** Se representa con el símbolo del acento circunflejo (^). Su clase léxica será *indireccion*.
- **Por Referencia.** Se representa con el símbolo ampersand una única vez (&). Su clase léxica será *param-Ref*.
- **Corchete de apertura.** Se representa con el símbolo del corchete de apertura “[”. Su clase léxica será *corcheteApert.*
- **Corchete de cierre.** Se representa con el símbolo del corchete de cierre “]”. Su clase léxica será *corcheteCierre*.
- **Punto.** Se representa con el símbolo punto (.). Su clase léxica será *punto*.

## 2.3. Especificación formal del léxico

### 2.3.1. Definiciones auxiliares.

$letra \longrightarrow [a - z, A - Z]$   
 $digitoPositivo \longrightarrow [1 - 9]$   
 $digito \longrightarrow digitoPositivo | 0$   
 $parteEntera \longrightarrow [\backslash +, \backslash -]? (\{ digitoPositivo \} \{ digito \}^* | 0)$   
 $parteDecimal \longrightarrow (\{ digito \}^* \{ digitoPositivo \} | 0)$   
 $parteExponencial \longrightarrow (e | E) parteEntera$

### 2.3.2. Definiciones léxicas.

$bool \longrightarrow (b | B) (o | O) (o | O) (l | L)$   
 $int \longrightarrow (i | I) (n | N) (t | T)$   
 $real \longrightarrow (r | R) (e | E) (a | A) (l | L)$   
 $string \longrightarrow (s | S) (t | T) (r | R) (i | I) (n | N) (g | G)$   
 $and \longrightarrow (a | A) (n | N) (d | D)$   
 $or \longrightarrow (o | O) (r | R)$   
 $not \longrightarrow (n | N) (o | O) (t | T)$   
 $true \longrightarrow (t | T) (r | R) (u | U) (e | E)$   
 $false \longrightarrow (f | F) (a | A) (l | L) (s | S) (e | E)$   
 $null \longrightarrow (n | N) (u | U) (l | L) (l | L)$   
 $proc \longrightarrow (p | P) (r | R) (o | O) (c | C)$   
 $if \longrightarrow (i | I) (f | F)$   
 $else \longrightarrow (e | E) (l | L) (s | S) (e | E)$   
 $while \longrightarrow (w | W) (h | H) (i | I) (l | L) (e | E)$   
 $struct \longrightarrow (s | S) (t | T) (r | R) (u | U) (c | C) (t | T)$   
 $new \longrightarrow (n | N) (e | E) (w | W)$   
 $delete \longrightarrow (d | D) (e | E) (l | L) (e | E) (t | T) (e | E)$   
 $read \longrightarrow (r | R) (e | E) (a | A) (d | D)$   
 $write \longrightarrow (w | W) (r | R) (i | I) (t | T) (e | E)$   
 $nl \longrightarrow (n | N) (l | L)$   
 $type \longrightarrow (t | T) (y | Y) (p | P) (e | E)$   
 $call \longrightarrow (c | C) (a | A) (l | L) (l | L)$   
 $literalEntero \longrightarrow \{ parteEntera \}$   
 $literalReal \longrightarrow \{ parteEntera \} \backslash \{ parteDecimal \} \{ parteExponencial \} \backslash \{ parteDecimal \} \{ parteExponencial \}$   
 $identificador \longrightarrow (\_ | letra) (letra | digito | \_)^*$   
 $literalCadena \longrightarrow " [ ^ " ] ^*"$   
 $suma \longrightarrow \backslash +$   
 $resta \longrightarrow -$   
 $mul \longrightarrow \backslash *$   
 $div \longrightarrow /$



$mod \rightarrow \%$   
 $menor \rightarrow <$   
 $mayor \rightarrow >$   
 $igual \rightarrow ==$   
 $menorIgual \rightarrow <=$   
 $mayorIgual \rightarrow >=$   
 $asig \rightarrow =$   
 $finalAsig \rightarrow \&\&$   
 $parenApert \rightarrow ($   
 $parenCierre \rightarrow )$   
 $llaveApert \rightarrow \{$   
 $llaveCierre \rightarrow \}$   
 $puntoComa \rightarrow ;$   
 $coma \rightarrow ,$   
 $punto \rightarrow .$   
 $arroba \rightarrow @$   
 $indireccion \rightarrow \backslash$   
 $paramRef \rightarrow \&$   
 $corcheteApert \rightarrow [$   
 $corcheteCierre \rightarrow ]$

### 2.3.3. Definiciones de cadenas ignorables.

$separador \rightarrow [ , \backslash t, \backslash r, \backslash b, \backslash n]$   
 $comentario \rightarrow \#\# [^(\backslash n | \mathbf{EOF})]^*$

# 3 | Tiny (0)

## 3.1. Especificación Sintáctica (Gramática)

Implementamos la gramática que define la especificación sintáctica del lenguaje Tiny0 empleando los patrones explicados en clase (Diseño descendente, Reutilización, Nivel de Abstracción Equilibrado, Opcionalidad, Variantes, Listas y Expresiones).

Para ello definimos primero la estructura básica de todo programa:

```
programa  $\rightarrow$  bloque
bloque  $\rightarrow$  {seccion_declaraciones_opt seccion_instrucciones_opt}
```

### 3.1.1. Declaraciones

```
seccion_declaraciones_opt  $\rightarrow$  seccion_declaraciones &&
seccion_declaraciones_opt  $\rightarrow$   $\epsilon$ 
seccion_declaraciones  $\rightarrow$  seccion_declaraciones ; declaracion
seccion_declaraciones  $\rightarrow$  declaracion
declaracion  $\rightarrow$  tipo_nombre
```

### 3.1.2. Tipos

```
tipo_nombre  $\rightarrow$  tipo_base identificador
tipo_base  $\rightarrow$  int
tipo_base  $\rightarrow$  real
tipo_base  $\rightarrow$  bool
```

### 3.1.3. Instrucciones

```
seccion_instrucciones_opt  $\rightarrow$  seccion_instrucciones
seccion_instrucciones_opt  $\rightarrow$   $\epsilon$ 
seccion_instrucciones  $\rightarrow$  lista_instrucciones
lista_instrucciones  $\rightarrow$  lista_instrucciones ; instruccion
lista_instrucciones  $\rightarrow$  instruccion
instruccion  $\rightarrow$  @ expresion
```

### 3.1.4. Expresiones

```
expresion  $\rightarrow$  E0
E0  $\rightarrow$  E1 = E0
E0  $\rightarrow$  E1
E1  $\rightarrow$  E1 op_relacional E2
E1  $\rightarrow$  E2
E2  $\rightarrow$  E2 + E3
E2  $\rightarrow$  E3 - E3
E2  $\rightarrow$  E3
E3  $\rightarrow$  E4 and E3
E3  $\rightarrow$  E4 or E4
E3  $\rightarrow$  E4
E4  $\rightarrow$  E4 op_mult E5
E4  $\rightarrow$  E5
E5  $\rightarrow$  - E5
E5  $\rightarrow$  not E5
```

$E5 \rightarrow E6$   
 $E6 \rightarrow \text{expresion\_basica}$   
 $E6 \rightarrow (E0)$   
 $\text{expresion\_basica} \rightarrow \text{literalEntero}$   
 $\text{expresion\_basica} \rightarrow \text{literalReal}$   
 $\text{expresion\_basica} \rightarrow \text{true}$   
 $\text{expresion\_basica} \rightarrow \text{false}$   
 $\text{expresion\_basica} \rightarrow \text{identificador}$

### 3.1.5. Operadores

$\text{op\_relacional} \rightarrow <$   
 $\text{op\_relacional} \rightarrow <=$   
 $\text{op\_relacional} \rightarrow >$   
 $\text{op\_relacional} \rightarrow >=$   
 $\text{op\_relacional} \rightarrow ==$   
 $\text{op\_relacional} \rightarrow !=$   
 $\text{op\_mult} \rightarrow *$   
 $\text{op\_mult} \rightarrow /$

## 3.2. Acondicionamiento

Acondicionamos la gramática definida en la sección anterior. Ésto, con el fin de implementar un analizador sintáctico descendente predictivo recursivo.

$\text{programa} \rightarrow \text{bloque}$   
 $\text{bloque} \rightarrow \{\text{seccion\_declaraciones\_opt seccion\_instrucciones\_opt}\}$

### 3.2.1. Declaraciones

$\text{seccion\_declaraciones\_opt} \rightarrow \text{seccion\_declaraciones} \&\&$   
 $\text{seccion\_declaraciones\_opt} \rightarrow \epsilon$   
 $\text{seccion\_declaraciones} \rightarrow \text{declaracion resto\_sd}$   
 $\text{resto\_sd} \rightarrow ; \text{declaracion resto\_sd}$   
 $\text{resto\_sd} \rightarrow \epsilon$   
 $\text{declaracion} \rightarrow \text{tipo\_nombre}$

### 3.2.2. Tipos

$\text{tipo\_nombre} \rightarrow \text{tipo\_base identificador}$   
 $\text{tipo\_base} \rightarrow \text{int}$   
 $\text{tipo\_base} \rightarrow \text{real}$   
 $\text{tipo\_base} \rightarrow \text{bool}$

### 3.2.3. Instrucciones

$\text{seccion\_instrucciones\_opt} \rightarrow \text{seccion\_instrucciones}$   
 $\text{seccion\_instrucciones\_opt} \rightarrow \epsilon$   
 $\text{seccion\_instrucciones} \rightarrow \text{lista\_instrucciones}$   
 $\text{lista\_instrucciones} \rightarrow \text{instruccion resto\_li}$   
 $\text{resto\_li} \rightarrow ; \text{instruccion resto\_li}$   
 $\text{resto\_li} \rightarrow \epsilon$   
 $\text{instruccion} \rightarrow @ \text{expresion}$

### 3.2.4. Expresiones

$expresion \rightarrow E0$   
 $E0 \rightarrow E1 \text{ resto\_} E0$   
 $resto\_E0 \rightarrow = E0$   
 $resto\_E0 \rightarrow \epsilon$   
 $E1 \rightarrow E2 \text{ resto\_} E1$   
 $resto\_E1 \rightarrow op\_relacional E2 \text{ resto\_} E1$   
 $resto\_E1 \rightarrow \epsilon$   
 $E2 \rightarrow E3 \text{ resto\_} E2\_F \text{ resto\_} E2\_R$   
 $resto\_E2\_R \rightarrow + E3 \text{ resto\_} E2\_R$   
 $resto\_E2\_R \rightarrow \epsilon$   
 $resto\_E2\_F \rightarrow - E3$   
 $resto\_E2\_F \rightarrow \epsilon$   
 $E3 \rightarrow E4 \text{ resto\_} E3$   
 $resto\_E3 \rightarrow \text{and } E3$   
 $resto\_E3 \rightarrow \text{or } E4$   
 $resto\_E3 \rightarrow \epsilon$   
 $E4 \rightarrow E5 \text{ resto\_} E4$   
 $resto\_E4 \rightarrow op\_mult E5 \text{ resto\_} E4$   
 $resto\_E4 \rightarrow \epsilon$   
 $E5 \rightarrow - E5$   
 $E5 \rightarrow \text{not } E5$   
 $E5 \rightarrow E6$   
 $E6 \rightarrow expresion\_basica$   
 $E6 \rightarrow (E0)$   
 $expresion\_basica \rightarrow \text{literalEntero}$   
 $expresion\_basica \rightarrow \text{literalReal}$   
 $expresion\_basica \rightarrow \text{true}$   
 $expresion\_basica \rightarrow \text{false}$   
 $expresion\_basica \rightarrow \text{identificador}$

### 3.2.5. Operadores

$op\_relacional \rightarrow <$   
 $op\_relacional \rightarrow <=$   
 $op\_relacional \rightarrow >$   
 $op\_relacional \rightarrow >=$   
 $op\_relacional \rightarrow ==$   
 $op\_relacional \rightarrow !=$   
 $op\_mult \rightarrow *$   
 $op\_mult \rightarrow /$

## 3.3. Directores

Directores de cada regla de la gramática acondicionada

### 3.3.1. Tabla de Reglas

Cuadro 3.3.1: Directores de las reglas de la gramática

Regla	Directores	Anulable
$programa \rightarrow bloque$	{	No
$bloque \rightarrow \{ seccion\_declaraciones\_opt seccion\_instrucciones\_opt \}$	{	No

Continúa en la siguiente página

Cuadro 3.3.1: Directores de las reglas de la gramática (Continuación)

Regla	Directores	Anulable
$seccion\_declaraciones\_opt \rightarrow seccion\_declaraciones \&\&$	<b>int real bool</b>	No
$seccion\_declaraciones\_opt \rightarrow \epsilon$		Sí
$seccion\_declaraciones \rightarrow declaracion\ resto\_sd$	<b>int real bool</b>	No
$resto\_sd \rightarrow ;\ declaracion\ resto\_sd$	<b>;</b>	No
$resto\_sd \rightarrow \epsilon$		Sí
$declaracion \rightarrow tipo\_nombre$	<b>int real bool</b>	No
$tipo\_nombre \rightarrow tipo\_base\ identificador$	<b>int real bool</b>	No
$tipo\_base \rightarrow \mathbf{int}$	<b>int</b>	No
$tipo\_base \rightarrow \mathbf{real}$	<b>real</b>	No
$tipo\_base \rightarrow \mathbf{bool}$	<b>bool</b>	No
$seccion\_instrucciones\_opt \rightarrow seccion\_instrucciones$	<b>@</b>	No
$seccion\_instrucciones\_opt \rightarrow \epsilon$		Sí
$seccion\_instrucciones \rightarrow lista\_instrucciones$	<b>@</b>	No
$lista\_instrucciones \rightarrow instruccion\ resto\_li$	<b>@</b>	No
$resto\_li \rightarrow ;\ instruccion\ resto\_li$	<b>;</b>	No
$resto\_li \rightarrow \epsilon$		Sí
$instruccion \rightarrow @\ expresion$	<b>@</b>	No
$expresion \rightarrow E0$	<b>- not literalReal literalEntero true false identificador (</b>	No
$E0 \rightarrow E1\ resto\_E0$	<b>- not literalReal literalEntero true false identificador (</b>	No
$resto\_E0 \rightarrow =\ E0$	<b>=</b>	No
$resto\_E0 \rightarrow \epsilon$		Sí
$E1 \rightarrow E2\ resto\_E1$	<b>- not literalReal literalEntero true false identificador (</b>	No
$resto\_E1 \rightarrow op\_relacional\ E2\ resto\_E1$	<b>&lt; &lt;= &gt; &gt;= == !=</b>	No
$resto\_E1 \rightarrow \epsilon$		Sí
$E2 \rightarrow E3\ resto\_E2\_F\ resto\_E2\_R$	<b>- not literalReal literalEntero true false identificador (</b>	No
$resto\_E2\_R \rightarrow +\ E3\ resto\_E2\_R$	<b>+</b>	No
$resto\_E2\_R \rightarrow \epsilon$		Sí
$resto\_E2\_F \rightarrow -\ E3$	<b>-</b>	No
$resto\_E2\_F \rightarrow \epsilon$		Sí

Continúa en la siguiente página

Cuadro 3.3.1: Directores de las reglas de la gramática (Continuación)

Regla	Directores	Anulable
$E3 \rightarrow E4 \text{ resto\_} E3$	- not literalReal literalEntero true false identificador (	No
$\text{resto\_} E3 \rightarrow \text{and } E3$	and	No
$\text{resto\_} E3 \rightarrow \text{or } E4$	or	No
$\text{resto\_} E3 \rightarrow \epsilon$		Sí
$E4 \rightarrow E5 \text{ resto\_} E4$	- not literalReal literalEntero true false identificador (	No
$\text{resto\_} E4 \rightarrow \text{op\_mult } E5 \text{ resto\_} E4$	* /	No
$\text{resto\_} E4 \rightarrow \epsilon$		Sí
$E5 \rightarrow - E5$	-	No
$E5 \rightarrow \text{not } E5$	not	No
$E5 \rightarrow E6$	literalReal litera- lEntero true false identificador (	No
$E6 \rightarrow \text{expresion\_basica}$	literalReal litera- lEntero true false identificador (	No
$E6 \rightarrow (E0)$	(	No
$\text{expresion\_basica} \rightarrow \text{literalEntero}$	literalEntero	No
$\text{expresion\_basica} \rightarrow \text{literalReal}$	literalReal	No
$\text{expresion\_basica} \rightarrow \text{true}$	true	No
$\text{expresion\_basica} \rightarrow \text{false}$	false	No
$\text{expresion\_basica} \rightarrow \text{identificador}$	identificador	No
$\text{op\_relacional} \rightarrow <$	<	No
$\text{op\_relacional} \rightarrow <=$	<=	No
$\text{op\_relacional} \rightarrow >$	>	No
$\text{op\_relacional} \rightarrow >=$	>=	No
$\text{op\_relacional} \rightarrow ==$	==	No
$\text{op\_relacional} \rightarrow !=$	!=	No
$\text{op\_mult} \rightarrow *$	*	No
$\text{op\_mult} \rightarrow /$	/	No

# 4 | Tiny

## 4.1. Especificación Sintáctica (Gramática)

Implementamos la gramática que define la especificación sintáctica del lenguaje Tiny empleando los patrones explicados en clase (Diseño descendente, Reutilización, Nivel de Abstracción Equilibrado, Opcionalidad, Variantes, Listas y Expresiones).

Para ello definimos primero la estructura básica de todo programa:

```
programa  $\rightarrow$  bloque
bloque  $\rightarrow$  {seccion_declaraciones_opt seccion_instrucciones_opt}
```

### 4.1.1. Declaraciones

```
seccion_declaraciones_opt  $\rightarrow$  seccion_declaraciones &&
seccion_declaraciones_opt  $\rightarrow$   $\epsilon$ 
seccion_declaraciones  $\rightarrow$  seccion_declaraciones ; declaracion
seccion_declaraciones  $\rightarrow$  declaracion
declaracion  $\rightarrow$  tipo_nombre
declaracion  $\rightarrow$  type tipo_nombre
declaracion  $\rightarrow$  proc identificador parametros_formales bloque
parametros_formales  $\rightarrow$  (lista_parametros_opt)
lista_parametros_opt  $\rightarrow$  lista_parametros
lista_parametros_opt  $\rightarrow$   $\epsilon$ 
lista_parametros  $\rightarrow$  lista_parametros , parametro
lista_parametros  $\rightarrow$  parametro
parametro  $\rightarrow$  tipo ref_opt identificador
ref_opt  $\rightarrow$  &
ref_opt  $\rightarrow$   $\epsilon$ 
```

### 4.1.2. Tipos

```
tipo_nombre  $\rightarrow$  tipo identificador
tipo  $\rightarrow$  tipo0
tipo0  $\rightarrow$  tipo0 [literalEntero]
tipo0  $\rightarrow$  tipo1
tipo1  $\rightarrow$  ^tipo1
tipo1  $\rightarrow$  tipo_base
tipo_base  $\rightarrow$  struct {lista_campos}
tipo_base  $\rightarrow$  int
tipo_base  $\rightarrow$  real
tipo_base  $\rightarrow$  bool
tipo_base  $\rightarrow$  string
tipo_base  $\rightarrow$  identificador
lista_campos  $\rightarrow$  lista_campos , tipo_nombre
lista_campos  $\rightarrow$  tipo_nombre
```

### 4.1.3. Instrucciones

```
seccion_instrucciones_opt  $\rightarrow$  seccion_instrucciones
seccion_instrucciones_opt  $\rightarrow$   $\epsilon$ 
seccion_instrucciones  $\rightarrow$  lista_instrucciones
lista_instrucciones  $\rightarrow$  lista_instrucciones ; instruccion
```

$lista\_instrucciones \rightarrow instruccion$   
 $instruccion \rightarrow @expresion$   
 $instruccion \rightarrow if\_ins$   
 $instruccion \rightarrow if\_ins\ else\_ins$   
 $instruccion \rightarrow \mathbf{while}\ exp\_bloque$   
 $instruccion \rightarrow \mathbf{read}\ expresion$   
 $instruccion \rightarrow \mathbf{write}\ expresion$   
 $instruccion \rightarrow \mathbf{nl}$   
 $instruccion \rightarrow \mathbf{new}\ expresion$   
 $instruccion \rightarrow \mathbf{delete}\ expresion$   
 $instruccion \rightarrow \mathbf{call\ identificador}\ parametros\_reales$   
 $instruccion \rightarrow bloque$   
 $if\_ins \rightarrow \mathbf{if}\ exp\_bloq$   
 $else\_ins \rightarrow \mathbf{else}\ bloque$   
 $exp\_bloq \rightarrow expresion\ bloque$   
 $parametros\_reales \rightarrow (lista\_expresiones\_opt)$   
 $lista\_expresiones\_opt \rightarrow lista\_expresiones$   
 $lista\_expresiones\_opt \rightarrow \epsilon$   
 $lista\_expresiones \rightarrow lista\_expresiones\ ,\ expresion$   
 $lista\_expresiones \rightarrow expresion$

#### 4.1.4. Expresiones

$expresion \rightarrow E0$   
 $E0 \rightarrow E1 = E0$   
 $E0 \rightarrow E1$   
 $E1 \rightarrow E1\ op\_relacional\ E2$   
 $E1 \rightarrow E2$   
 $E2 \rightarrow E2 + E3$   
 $E2 \rightarrow E3 - E3$   
 $E2 \rightarrow E3$   
 $E3 \rightarrow E4\ \mathbf{and}\ E3$   
 $E3 \rightarrow E4\ \mathbf{or}\ E4$   
 $E3 \rightarrow E4$   
 $E4 \rightarrow E4\ op\_mult\ E5$   
 $E4 \rightarrow E5$   
 $E5 \rightarrow -\ E5$   
 $E5 \rightarrow \mathbf{not}\ E5$   
 $E5 \rightarrow E6$   
 $E6 \rightarrow E6\ op\_dirs$   
 $E6 \rightarrow E7$   
 $E7 \rightarrow expresion\_basica$   
 $E7 \rightarrow (E0)$   
 $expresion\_basica \rightarrow \mathbf{literalEntero}$   
 $expresion\_basica \rightarrow \mathbf{literalReal}$   
 $expresion\_basica \rightarrow \mathbf{true}$   
 $expresion\_basica \rightarrow \mathbf{false}$   
 $expresion\_basica \rightarrow \mathbf{literalCadena}$   
 $expresion\_basica \rightarrow \mathbf{identificador}$   
 $expresion\_basica \rightarrow \mathbf{null}$

#### 4.1.5. Operadores

$op\_relacional \rightarrow <$   
 $op\_relacional \rightarrow <=$   
 $op\_relacional \rightarrow >$   
 $op\_relacional \rightarrow >=$   
 $op\_relacional \rightarrow ==$   
 $op\_relacional \rightarrow !=$



$op\_mult \rightarrow *$   
 $op\_mult \rightarrow /$   
 $op\_mult \rightarrow \%$   
 $op\_dirs \rightarrow [expresion]$   
 $op\_dirs \rightarrow \texttt{.identificador}$   
 $op\_dirs \rightarrow \wedge$

## 4.2. Acondicionamiento

Acondicionamos la gramática definida en la sección anterior. Ésto, con el fin de implementar un analizador sintáctico descendente predictivo recursivo.

$programa \rightarrow bloque$   
 $bloque \rightarrow \{seccion\_declaraciones\_opt\} seccion\_instrucciones\_opt\}$

### 4.2.1. Declaraciones

$seccion\_declaraciones\_opt \rightarrow seccion\_declaraciones \&\&$   
 $seccion\_declaraciones\_opt \rightarrow \epsilon$   
 $seccion\_declaraciones \rightarrow declaracion\ resto\_sd$   
 $resto\_sd \rightarrow ;\ declaracion\ resto\_sd$   
 $resto\_sd \rightarrow \epsilon$   
 $declaracion \rightarrow tipo\_nombre$   
 $declaracion \rightarrow \texttt{type}\ tipo\_nombre$   
 $declaracion \rightarrow \texttt{proc identificador}\ parametros\_formales\ bloque$   
 $parametros\_formales \rightarrow (lista\_parametros\_opt)$   
 $lista\_parametros\_opt \rightarrow lista\_parametros$   
 $lista\_parametros\_opt \rightarrow \epsilon$   
 $lista\_parametros \rightarrow parametro\ resto\_lp$   
 $resto\_lp \rightarrow ,\ parametro\ resto\_lp$   
 $resto\_lp \rightarrow \epsilon$   
 $parametro \rightarrow tipo\ ref\_opt\ identificador$   
 $ref\_opt \rightarrow \&$   
 $ref\_opt \rightarrow \epsilon$

### 4.2.2. Tipos

$tipo\_nombre \rightarrow tipo\ identificador$   
 $tipo \rightarrow tipo0$   
 $tipo0 \rightarrow tipo1\ resto\_tipo0$   
 $resto\_tipo0 \rightarrow [\texttt{literalEntero}]\ resto\_tipo0$   
 $resto\_tipo0 \rightarrow \epsilon$   
 $tipo1 \rightarrow \wedge tipo1$   
 $tipo1 \rightarrow tipo\_base$   
 $tipo\_base \rightarrow \texttt{struct}\ \{lista\_campos\}$   
 $tipo\_base \rightarrow \texttt{int}$   
 $tipo\_base \rightarrow \texttt{real}$   
 $tipo\_base \rightarrow \texttt{bool}$   
 $tipo\_base \rightarrow \texttt{string}$   
 $tipo\_base \rightarrow identificador$   
 $lista\_campos \rightarrow tipo\_nombre\ resto\_lc$   
 $resto\_lc \rightarrow ,\ tipo\_nombre\ resto\_lc$   
 $resto\_lc \rightarrow \epsilon$

### 4.2.3. Instrucciones

$seccion\_instrucciones\_opt \rightarrow seccion\_instrucciones$   
 $seccion\_instrucciones\_opt \rightarrow \epsilon$   
 $seccion\_instrucciones \rightarrow lista\_instrucciones$   
 $lista\_instrucciones \rightarrow instruccion\ resto\_li$   
 $resto\_li \rightarrow ;\ instruccion\ resto\_li$   
 $resto\_li \rightarrow \epsilon$   
 $instruccion \rightarrow @\ expresion$   
 $instruccion \rightarrow if\_ins\ resto\_ii$   
 $resto\_ii \rightarrow else\_ins$   
 $resto\_ii \rightarrow \epsilon$   
 $instruccion \rightarrow \textbf{while}\ exp\_bloque$   
 $instruccion \rightarrow \textbf{read}\ expresion$   
 $instruccion \rightarrow \textbf{write}\ expresion$   
 $instruccion \rightarrow \textbf{nl}$   
 $instruccion \rightarrow \textbf{new}\ expresion$   
 $instruccion \rightarrow \textbf{delete}\ expresion$   
 $instruccion \rightarrow \textbf{call}\ identificador\ parametros\_reales$   
 $instruccion \rightarrow bloque$   
 $if\_ins \rightarrow \textbf{if}\ exp\_bloq$   
 $else\_ins \rightarrow \textbf{else}\ bloque$   
 $exp\_bloq \rightarrow expresion\ bloque$   
 $parametros\_reales \rightarrow (lista\_expresiones\_opt)$   
 $lista\_expresiones\_opt \rightarrow lista\_expresiones$   
 $lista\_expresiones\_opt \rightarrow \epsilon$   
 $lista\_expresiones \rightarrow expresion\ resto\_le$   
 $resto\_le \rightarrow ,\ expresion\ resto\_le$   
 $resto\_le \rightarrow \epsilon$

### 4.2.4. Expresiones

$expresion \rightarrow E0$   
 $E0 \rightarrow E1\ resto\_E0$   
 $resto\_E0 \rightarrow =\ E0$   
 $resto\_E0 \rightarrow \epsilon$   
 $E1 \rightarrow E2\ resto\_E1$   
 $resto\_E1 \rightarrow op\_relacional\ E2\ resto\_E1$   
 $resto\_E1 \rightarrow \epsilon$   
 $E2 \rightarrow E3\ resto\_E2\_F\ resto\_E2\_R$   
 $resto\_E2\_R \rightarrow +\ E3\ resto\_E2\_R$   
 $resto\_E2\_R \rightarrow \epsilon$   
 $resto\_E2\_F \rightarrow -\ E3$   
 $resto\_E2\_F \rightarrow \epsilon$   
 $E3 \rightarrow E4\ resto\_E3$   
 $resto\_E3 \rightarrow \textbf{and}\ E3$   
 $resto\_E3 \rightarrow \textbf{or}\ E4$   
 $resto\_E3 \rightarrow \epsilon$   
 $E4 \rightarrow E5\ resto\_E4$   
 $resto\_E4 \rightarrow op\_mult\ E5\ resto\_E4$   
 $resto\_E4 \rightarrow \epsilon$   
 $E5 \rightarrow -\ E5$   
 $E5 \rightarrow \textbf{not}\ E5$   
 $E5 \rightarrow E6$   
 $E6 \rightarrow E7\ resto\_E6$   
 $resto\_E6 \rightarrow op\_dirs\ resto\_E6$   
 $resto\_E6 \rightarrow \epsilon$   
 $E7 \rightarrow expresion\_basica$   
 $E7 \rightarrow (E0)$   
 $expresion\_basica \rightarrow \textbf{literalEntero}$

*expresion\_basica*  $\rightarrow$  **literalReal**  
*expresion\_basica*  $\rightarrow$  **true**  
*expresion\_basica*  $\rightarrow$  **false**  
*expresion\_basica*  $\rightarrow$  **literalCadena**  
*expresion\_basica*  $\rightarrow$  **identificador**  
*expresion\_basica*  $\rightarrow$  **null**

#### 4.2.5. Operadores

*op\_relacional*  $\rightarrow$  <  
*op\_relacional*  $\rightarrow$  <=  
*op\_relacional*  $\rightarrow$  >  
*op\_relacional*  $\rightarrow$  >=  
*op\_relacional*  $\rightarrow$  ==  
*op\_relacional*  $\rightarrow$  !=  
*op\_mult*  $\rightarrow$  \*  
*op\_mult*  $\rightarrow$  /  
*op\_mult*  $\rightarrow$  %  
*op\_dirs*  $\rightarrow$  [*expresion*]  
*op\_dirs*  $\rightarrow$  .**identificador**  
*op\_dirs*  $\rightarrow$  ^

# Índice de figuras

1.4.1. AFD del analizador léxico de Tiny (0) . . . . .	4
--	---

# Índice de cuadros

3.3.1.Directores de las reglas de la gramática . . . . .	11
--	----