**Practica 2:**

**Generación del analizador léxico con JLex**

**Diseño de la sintaxis**

**Análisis sintáctico descendente**

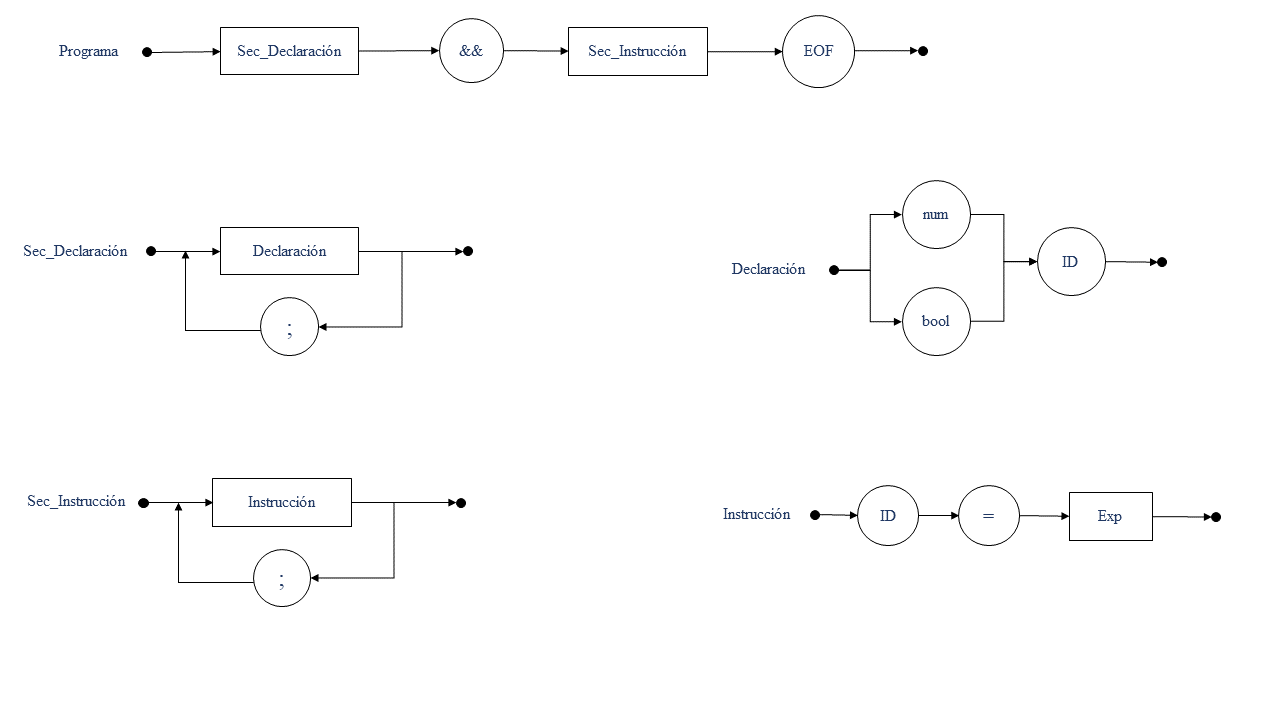
**Grupo 11:**

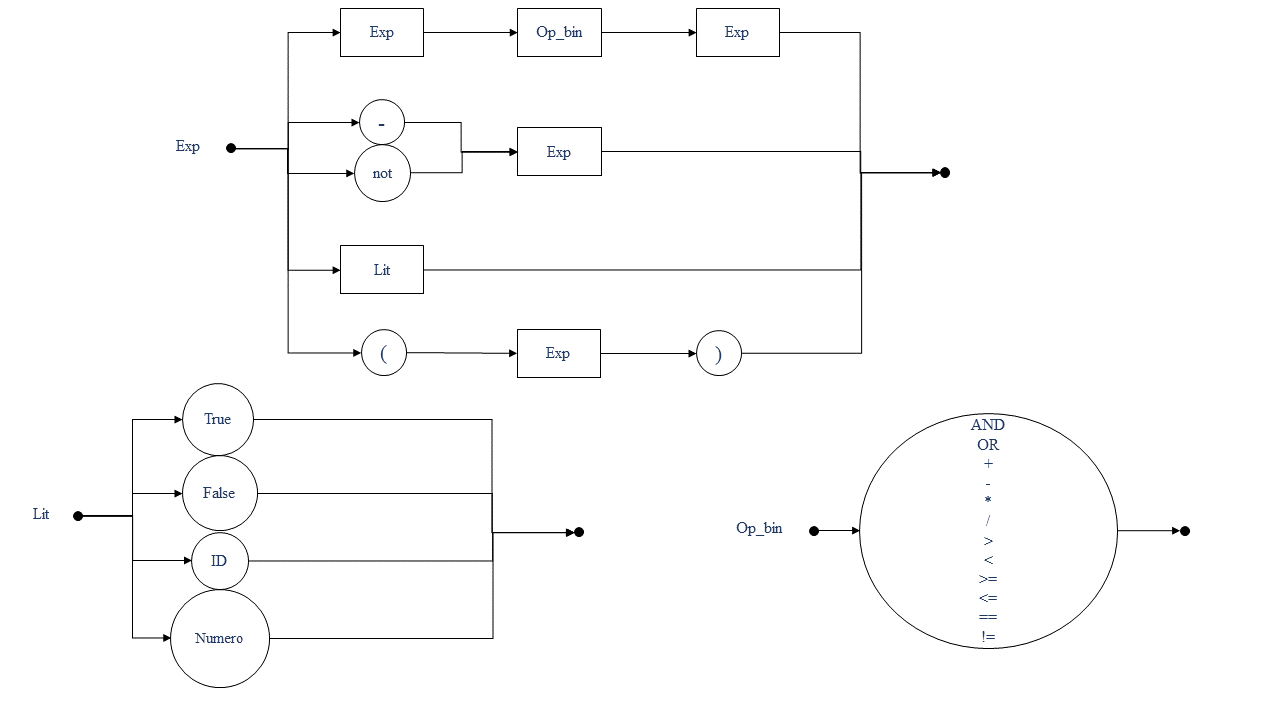
**Youssef El Faqir El Rhazoui**

**Enrique Ávila Rodríguez**

1. **Conceptualización mediante diagramas**

Pasamos a describir el lenguaje mediante los diagramas.





1. **Gramática Incontextual**

Ahora pasamos a especificar la gramática, teniendo en cuenta la asociatividad y prioridad de los operadores dados por la siguiente tabla.

|  |  |  |
| --- | --- | --- |
| Operadores | Prioridad | Asociatividad |
| (binario) +, - | 0 | * Asoc. Izq |
| And, or | 1 | * And asoc derch. * Or no asoc |
| ==, ¡=, <, <=, >, >= | 2 | * No asoc |
| \*, / | 3 | * Asoc izq |
| - (unario), not | 4 | * - asoc * Not no asoc |

Programa 🡪 Sec\_Declaración && Sec\_Instrucción EOF

Sec\_Declaración 🡪 Declaración

Sec\_Declaración 🡪 Declaración ; Sec\_Declaración

Declaración 🡪 Tipo ID

Tipo 🡪 num

Tipo 🡪 bool

Sec\_Instrucción 🡪 Instrucción

Sec\_Instrucción 🡪 Instrucción ; Sec\_Instrucción

Instrucción 🡪 ID = Exp

Exp 🡪 Exp0

Exp0 🡪 Exp0 Op0 Exp1

Exp0 🡪 Exp1

Op0 🡪 +

Op0 🡪 -

Exp1 🡪 Exp2 and Exp1

Exp1 🡪 Exp2 or Exp2

Exp1 🡪 Exp2

Exp2 🡪 Exp3 Op2 Exp3

Exp2 🡪 Exp3

Op2 🡪 ¡= **|** == **|** < **|** <= **|** > **|** >=

Exp3 🡪 Exp3 Op3 Exp4

Exp3 🡪 Exp4

Op3 🡪 \*

Op3 🡪 /

Exp4 🡪 - Exp4

Exp4 🡪 not Exp5

Exp4 🡪 Exp5

Exp5 🡪 Lit

Exp5 🡪 ( Exp0 )

Lit 🡪 ID

Lit 🡪 True

Lit 🡪 False

Lit 🡪 Numero

1. **Transformaciones para gramática LL1**

Para poder hacer un análisis descendente práctico, la gramática anterior no nos vale debido a que presenta problemas de recursión a la izquierda y de factor común.

**Resolución de Factor Común**

**(1)**

Exp1 🡪 Exp2 and Exp1

Exp1 🡪 Exp2 or Exp2

Exp1 🡪 Exp2

Exp1 🡪 Exp2 Exp1p

Exp1p 🡪 and Exp1

Exp1p 🡪 or Exp2

Exp1p 🡪 ε

**(2)**

Exp2 🡪 Exp3 Op2 Exp3

Exp2 🡪 Exp3

Exp2 🡪 Exp3 Exp2p

Exp2p 🡪 Op2 Exp3

Exp2p 🡪 ε

**(3)**

Sec\_Declaración 🡪 Declaración

Sec\_Declaración 🡪 Declaración ; Sec\_Declaración

Sec\_Declaración 🡪 Declaración Sec\_Declaración\_p

Sec\_Declaración\_p 🡪 ; Sec\_Declaración

Sec\_Declaración\_p 🡪 ε

**(4)**

Sec\_Instrucción 🡪 Instrucción

Sec\_Instrucción 🡪 Instrucción ; Sec\_Instrucción

Sec\_Instrucción 🡪 Instrucción Sec\_Instrucción\_p

Sec\_Instrucción\_p 🡪 ; Sec\_Instrucción

Sec\_Instrucción\_p 🡪 ε

**Resolución de recursión a la izq**

**(5)**

Exp0 🡪 Exp0 Op0 Exp1

Exp0 🡪 Exp1

Exp0 🡪 Exp1 Exp0p

Exp0p 🡪 Op0 Exp1 Exp0p

Exp0p 🡪 ε

**(6)**

Exp3 🡪 Exp3 Op3 Exp4

Exp3 🡪 Exp4

Exp3 🡪 Exp4 Exp3p

Exp3p 🡪 Op3 Exp4 Exp3p

Exp3p 🡪 ε

**Con lo cual, la gramática LL1 resultante es:**

Programa 🡪 Sec\_Declaración && Sec\_Instrucción EOF

Sec\_Declaración 🡪 Declaración Sec\_Declaración\_p

Sec\_Declaración\_p 🡪 ; Sec\_Declaración

Sec\_Declaración\_p 🡪 ε

Declaración 🡪 Tipo ID

Tipo 🡪 num

Tipo 🡪 bool

Sec\_Instrucción 🡪 Instrucción Sec\_Instrucción\_p

Sec\_Instrucción\_p 🡪 ; Sec\_Instrucción

Sec\_Instrucción\_p 🡪 ε

Instrucción 🡪 ID = Exp

Exp 🡪 Exp0

Exp0 🡪 Exp1 Exp0p

Exp0p 🡪 Op0 Exp1 Exp0p

Exp0p 🡪 ε

Op0 🡪 +

Op0 🡪 -

Exp1 🡪 Exp2 Exp1p

Exp1p 🡪 and Exp1

Exp1p 🡪 or Exp2

Exp1p 🡪 ε

Exp2 🡪 Exp3 Exp2p

Exp2p 🡪 Op2 Exp3

Exp2p 🡪 ε

Op2 🡪 ¡= **|** == **|** < **|** <= **|** > **|** >=

Exp3 🡪 Exp4 Exp3p

Exp3p 🡪 Op3 Exp4 Exp3p

Exp3p 🡪 ε

Op3 🡪 \*

Op3 🡪 /

Exp4 🡪 - Exp4

Exp4 🡪 not Exp5

Exp4 🡪 Exp5

Exp5 🡪 Lit

Exp5 🡪 ( Exp0 )

Lit 🡪 ID

Lit 🡪 True

Lit 🡪 False

Lit 🡪 Numero

1. **Primeros, siguientes y directores**

Nos hemos apoyado en [proletool](https://portal.esi.uclm.es/proletool) para generarlos, adjuntamos el archivo “proletool\_file.txt” usado en esta herramienta.

|  |  |
| --- | --- |
| **No terminal** | **Primeros** |
| Instruccion | ID |
| Sec\_Instruccion | ID |
| Sec\_Declaracion | bool num |
| Op0 | + - |
| Op2 | < <= >= > == != |
| Programa | bool num |
| Declaracion | bool num |
| Exp0 | Numero not ( True ID False - |
| Tipo | bool num |
| Op3 | \* / |
| Exp2 | Numero not ( True ID False - |
| Exp1 | Numero not ( True ID False - |
| Sec\_Instruccion\_p | ; ε |
| Exp4 | Numero not ( True ID False - |
| Lit | Numero True ID False |
| Exp3 | Numero not ( True ID False - |
| Exp1p | or and ε |
| Exp0p | + - ε |
| Exp5 | Numero ( True ID False |
| Sec\_Declaracion\_p | ; ε |
| Exp | Numero not ( True ID False - |
| Exp3p | \* / ε |
| Exp2p | < <= >= > == != ε |

|  |  |
| --- | --- |
| **No terminal** | **Siguientes** |
| Instruccion | ; EOF |
| Sec\_Instruccion | EOF |
| Sec\_Declaracion | && |
| Op0 | Numero not ( True ID False - |
| Op2 | Numero not ( True ID False - |
| Programa | EOF |
| Declaracion | && ; |
| Exp0 | ) ; EOF |
| Tipo | ID |
| Op3 | Numero not ( True ID False - |
| Exp2 | or and ) + ; - EOF |
| Exp1 | ) + ; - EOF |
| Sec\_Instruccion\_p | EOF |
| Exp4 | and or < <= >= > == != ) \* + - / ; EOF |
| Lit | and or < <= >= > == != ) \* + - / ; EOF |
| Exp3 | and or < <= >= > == != ) + - ; EOF |
| Exp1p | ) + ; - EOF |
| Exp0p | ) ; EOF |
| Exp5 | and or < <= >= > == != ) \* + - / ; EOF |
| Sec\_Declaracion\_p | && |
| Exp | ; EOF |
| Exp3p | and or < <= >= > == != ) + - ; EOF |
| Exp2p | or and ) + ; - EOF |

|  |  |
| --- | --- |
| **No terminal** | **Directores** |
| Programa 🡪 Sec\_Declaración && Sec\_Instrucción EOF | bool num |
| Sec\_Declaración 🡪 Declaración Sec\_Declaración\_p | bool num |
| Sec\_Declaración\_p 🡪 ; Sec\_Declaración | ; |
| Sec\_Declaración\_p 🡪 ε | && |
| Declaración 🡪 Tipo ID | bool num |
| Tipo -> num | num |
| Tipo -> bool | bool |
| Sec\_Instrucción -> Instrucción Sec\_Instrucción\_p | ID |
| Sec\_Instrucción\_p -> ; Sec\_Instrucción | ; |
| Sec\_Instrucción\_p -> ε | EOF |
| Instrucción -> ID = Exp | ID |
| Exp -> Exp0 | Numero not ( True ID False - |
| Exp0 -> Exp1 Exp0p | Numero not ( True ID False - |
| Exp0p -> Op0 Exp1 Exp0p | + - |
| Exp0p -> ε | ) ; EOF |
| Op0 -> + | + |
| Op0 -> - | - |
| Exp1 -> Exp2 Exp1p | Numero not ( True ID False - |
| Exp1p -> and Exp1 | and |
| Exp1p -> or Exp2 | or |
| Exp1p -> ε | ) + ; - EOF |
| Exp2 -> Exp3 Exp2p | Numero not ( True ID False - |
| Exp2p -> Op2 Exp3 | < <= >= > == != |
| Exp2p -> ε | or and ) + ; - EOF |
| Op2 -> < | < |
| Op2 -> <= | <= |
| Op2 -> >= | >= |
| Op2 -> > | > |
| Op2 -> == | == |
| Op2 -> != | != |
| Exp3 -> Exp4 Exp3p | Numero not ( True ID False - |
| Exp3p -> Op3 Exp4 Exp3p | \* / |
| Exp3p -> ε | and or < <= >= > == != ) + - ; EOF |
| Op3 -> \* | \* |
| Op3 -> / | / |
| Exp4 -> - Exp4 | - |
| Exp4 -> not Exp5 | not |
| Exp4 -> Exp5 | Numero ( True ID False |
| Exp5 -> Lit | Numero True ID False |
| Exp5 -> ( Exp0 ) | ( |
| Lit -> ID | ID |
| Lit -> True | True |
| Lit -> False | False |
| Lit -> Numero | Numero |