**DAVID DE FRUTOS ZAFRA**

**PABLO GARCÍA GARCÍA**

**GRUPO 119**

|  |  |
| --- | --- |
| ANALIZADOR LÉXICO Y TABLA DE SÍMBOLOS  SEGUNDA ENTREGA - PDL | cONTENIDO  - Diseño del Analizador Sintáctico (gramática, demostración de que la gramática es adecuada para el método de Análisis Sintáctico asignado y tabla LL  - Anexo con 6 casos de prueba para mostrar el funcionamiento del Procesador, la mitad correctos y la mitad con errores sintácticos. Añadiendo el uso de la herramienta VASt . |

**DISEÑO DE GRAMÁTICA**

P → BP | FP |

B → let T id; | if G| S

T → int | string | boolean

G → (E) S | { C } O

C → BC |

O → else { C } |

S → id W | print (E); | input (id);| return X;

W → -=E; | =E; | (L);

X → E |

L → EQ |

Q → , EQ |

F → function id H (A) { C }

H → T |

A → T id K |

K → , T id K |

E → RE’

E’ → &&RE’ |

R → UR’

R’ → <UR’ | >UR’ |

U → VU’

U’ → +VU’ | -VU’ |

V → id D | (E) | entero | cadena

D → (L) |

Hemos comprobado que nuestra gramática cumple las propiedades de la gramática LL:

Texto, Carta

Descripción generada automáticamente

* Hemos eliminado la recursividad por la izquierda transformando las reglas desde la gramática sugerida que suponían un problema. Como se puede ver en la imagen
* No es ambigua para ninguna regla
* Esta factorizada dado que ningún consecuente de dos o más reglas de un No terminal comienza igual.

**DISEÑO TABLA LL**



1. P -> B P
2. P -> F P
3. P -> lambda
4. B -> let T id ;
5. B -> if G
6. B -> S
7. T -> int
8. T -> string
9. T -> boolean
10. G -> ( E ) S
11. G -> { C } O
12. C -> B C
13. C -> lambda
14. O -> else { C }
15. O -> lambda
16. S -> id W
17. S -> print ( E ) ;
18. S -> input ( id ) ;
19. S -> return X ;
20. W -> - = E ;
21. W -> = E ;
22. W -> ( L ) ;
23. X -> E
24. X -> lambda
25. L -> E Q
26. L -> lambda
27. Q -> , E Q
28. Q -> lambda
29. F -> function id H ( A ) { C }
30. H -> T
31. H -> lambda
32. A -> T id K
33. A -> lambda
34. K -> , T id K
35. K -> lambda
36. E -> R E\_
37. E\_ -> && R E\_
38. E\_ -> lambda
39. R -> U R\_
40. R\_ -> < U R\_
41. R\_ -> > U R\_
42. R\_ -> lambda
43. U -> V U\_
44. U\_ -> + V U\_
45. U\_ -> - V U\_
46. U\_ -> lambda
47. V -> id D
48. V -> ( E )
49. V -> entero
50. V -> cadena
51. D -> ( L )
52. D -> lambda