**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 (E) G| S

T → int | string | boolean

G → 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 (E) G
6. B -> S
7. T -> int
8. T -> string
9. T -> boolean
10. G -> 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

**PRUEBAS**

***PRUEBA 1****:* **correcta**

let boolean booleano;

function bisiesto boolean (int a)

{ let string bis;

print ('Es bisiesto?');

input(bis);

return ((a + 4 > 0));

}

print (4-5+77);

function dias int (int m, int a)

{

let int dd;

print ('di cuantos dias tiene el mes ');

print (m);

input(dd);

if (bisiesto(a)) dd = dd - 1;

return dd;

}

function esFechaCorrecta boolean (int d, int m, int a)

{

return m<1 && m>12 && d<1 && d > dias (m, a);

}

function demo ()

{

if (esFechaCorrecta(22, 22, 2022)) print (9999);

return;

}

let int a\_1\_b\_2;

demo();

PARSE:

Descendente 1 4 9 2 29 30 9 32 7 35 12 4 8 12 6 17 36 39 43 50 46 42 38 12 6 18 12 6 19 23 36 39 43 48 36 39 43 48 36 39 43 47 52 44 49 46 41 43 49 46 42 38 46 42 38 46 42 38 13 1 6 17 36 39 43 49 45 49 44 49 46 42 38 2 29 30 7 32 7 34 7 35 12 4 7 12 6 17 36 39 43 50 46 42 38 12 6 17 36 39 43 47 52 46 42 38 12 6 18 12 5 36 39 43 47 51 25 36 39 43 47 52 46 42 38 28 46 42 38 10 16 21 36 39 43 47 52 45 49 46 42 38 12 6 19 23 36 39 43 47 52 46 42 38 13 2 29 30 9 32 7 34 7 34 7 35 12 6 19 23 36 39 43 47 52 46 40 43 49 46 42 37 39 43 47 52 46 41 43 49 46 42 37 39 43 47 52 46 40 43 49 46 42 37 39 43 47 52 46 41 43 47 51 25 36 39 43 47 52 46 42 38 27 36 39 43 47 52 46 42 38 28 46 42 38 13 2 29 31 33 12 5 36 39 43 47 51 25 36 39 43 49 46 42 38 27 36 39 43 49 46 42 38 27 36 39 43 49 46 42 38 28 46 42 38 10 17 36 39 43 49 46 42 38 12 6 19 24 13 1 4 7 1 6 16 22 26 3

ÁRBOL VAST:

***PRUEBA 2****:* **correcta**

let int z;

let boolean boolean\_1;

let int x;

let string ss;

let int xx;

let boolean boolean\_2;

function f1 int(int f1, boolean b1)

{

print(ss);

x = xx+f1;

boolean\_1 = boolean\_1&& boolean\_2;

return (01234);

}

function f2 boolean( int f2 , boolean b1 )

{

input (y);

print ((4+5+77+(088-f2)));

return (boolean\_1&&boolean\_2&&b1);

}

x =

x + 6

- z

+ 1

+ (2

+ y

- 6)

;

print (f1 (x, f2 (3, boolean\_2)));

PARSE:

Descendente 1 4 7 1 4 9 1 4 7 1 4 8 1 4 7 1 4 9 2 29 30 7 32 7 34 9 35 12 6 17 36 39 43 47 52 46 42 38 12 6 16 21 36 39 43 47 52 44 47 52 46 42 38 12 6 16 21 36 39 43 47 52 46 42 37 39 43 47 52 46 42 38 12 6 19 23 36 39 43 48 36 39 43 49 46 42 38 46 42 38 13 2 29 30 9 32 7 34 9 35 12 6 18 12 6 17 36 39 43 48 36 39 43 49 44 49 44 49 44 48 36 39 43 49 45 47 52 46 42 38 46 42 38 46 42 38 12 6 19 23 36 39 43 48 36 39 43 47 52 46 42 37 39 43 47 52 46 42 37 39 43 47 52 46 42 38 46 42 38 13 1 6 16 21 36 39 43 47 52 44 49 45 47 52 44 49 44 48 36 39 43 49 44 47 52 45 49 46 42 38 46 42 38 1 6 17 36 39 43 47 51 25 36 39 43 47 52 46 42 38 27 36 39 43 47 51 25 36 39 43 49 46 42 38 27 36 39 43 47 52 46 42 38 28 46 42 38 28 46 42 38 3

ÁRBOL VAST:

***PRUEBA 3****:* **correcta**

let string texto;

function pideTexto ()

{

print ('Introduce un texto');

input (texto);

}

function alert (string msg)

{

print ('Texto introducido:');

print (msg);

}

pideTexto();

alert

(texto);

PARSE:

Descendente 1 4 8 2 29 31 33 12 6 17 36 39 43 50 46 42 38 12 6 18 13 2 29 31 32 8 35 12 6 17 36 39 43 50 46 42 38 12 6 17 36 39 43 47 52 46 42 38 13 1 6 16 22 26 1 6 16 22 25 36 39 43 47 52 46 42 38 28 3

ÁRBOL VAST:

***PRUEBA 4****:* ***in*correcta**

let int a ;

let int b ;

function operacion

int (int num1\_, int num2\_ //Error en esta linea

{

return number;

}

Error sintactico: se encuentra el token kAbierta cuando deberia aparecer el token pCerrado

***PRUEBA 5****:* ***in*correcta**

let int ; //Error en esta linea

let boolean b;

let int z;

input (x);

print (x);

input (z);

print (x+z);

b=x>z;if (b)

x =

x - 6

- z

- 1

- (2

+ y

- 7);

Error sintactico: se encuentra el token puntComa cuando deberia aparecer el token id

***PRUEBA 6****:* ***in*correcta**

let boolean b;let int x;

input (x);

print (x);

input (z);

print (x-z);

b=x>z;if (b //Error en esta linea

x =

x + 6

+ z

+ 1

+ (2

- y

- 7);

Error sintactico: se encuentra el token id cuando deberia aparecer el token pCerrado