**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**

**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:

* P (1)
  + B (4)
    - let
    - T (9)
      * boolean
    - id
    - ;
  + P (2)
    - F (29)
      * function
      * id
      * H (30)
        + T (9)

boolean

* + - * (
      * A (32)
        + T (7)

int

* + - * + id
        + K (35)

lambda

* + - * )
      * {
      * C (12)
        + B (4)

let

T (8)

string

id

;

* + - * + C (12)

B (6)

S (17)

print

(

E (36)

R (39)

* U (43)
  + V (50)
    - cadena
  + U\_ (46)
    - lambda
* R\_ (42)
  + lambda

E\_ (38)

* lambda

)

;

C (12)

B (6)

S (18)

input

(

id

)

;

C (12)

B (6)

S (19)

* return
* X (23)
  + E (36)
    - R (39)
      * U (43)
        + V (48)

(

E (36)

R (39)

U (43)

V (48)

* (
* E (36)
  + R (39)
    - U (43)
      * V (47)
        + id
        + D (52)

lambda

* + - * U\_ (44)
        + +
        + V (49)

entero

* + - * + U\_ (46)

lambda

* + - R\_ (41)
      * >
      * U (43)
        + V (49)

entero

* + - * + U\_ (46)

lambda

* + - * R\_ (42)
        + lambda
  + E\_ (38)
    - lambda
* )

U\_ (46)

* lambda

R\_ (42)

lambda

E\_ (38)

lambda

)

* + - * + U\_ (46)

lambda

* + - * R\_ (42)
        + lambda
    - E\_ (38)
      * lambda
* ;

C (13)

lambda

* + - * }
    - P (1)
      * B (6)
        + S (17)

print

(

E (36)

R (39)

U (43)

V (49)

* entero

U\_ (45)

* -
* V (49)
  + entero
* U\_ (44)
  + +
  + V (49)
    - entero
  + U\_ (46)
    - lambda

R\_ (42)

lambda

E\_ (38)

lambda

)

;

* + - * P (2)
        + F (29)

function

id

H (30)

T (7)

int

(

A (32)

T (7)

int

id

K (34)

,

T (7)

int

id

K (35)

lambda

)

{

C (12)

B (4)

let

T (7)

int

id

;

C (12)

B (6)

S (17)

* print
* (
* E (36)
  + R (39)
    - U (43)
      * V (50)
        + cadena
      * U\_ (46)
        + lambda
    - R\_ (42)
      * lambda
  + E\_ (38)
    - lambda
* )
* ;

C (12)

B (6)

* S (17)
  + print
  + (
  + E (36)
    - R (39)
      * U (43)
        + V (47)

id

D (52)

lambda

* + - * + U\_ (46)

lambda

* + - * R\_ (42)
        + lambda
    - E\_ (38)
      * lambda
  + )
  + ;

C (12)

* B (6)
  + S (18)
    - input
    - (
    - id
    - )
    - ;
* C (12)
  + B (5)
    - if
    - (
    - E (36)
      * R (39)
        + U (43)

V (47)

id

D (51)

(

L (25)

E (36)

* R (39)
  + U (43)
    - V (47)
      * id
      * D (52)
        + lambda
    - U\_ (46)
      * lambda
  + R\_ (42)
    - lambda
* E\_ (38)
  + lambda

Q (28)

* lambda

)

U\_ (46)

lambda

* + - * + R\_ (42)

lambda

* + - * E\_ (38)
        + lambda
    - )
    - G (10)
      * S (16)
        + id
        + W (21)

=

E (36)

R (39)

U (43)

V (47)

* id
* D (52)
  + lambda

U\_ (45)

* -
* V (49)
  + entero
* U\_ (46)
  + lambda

R\_ (42)

lambda

E\_ (38)

lambda

;

* + C (12)
    - B (6)
      * S (19)
        + return
        + X (23)

E (36)

R (39)

U (43)

V (47)

* id
* D (52)
  + lambda

U\_ (46)

* lambda

R\_ (42)

lambda

E\_ (38)

lambda

* + - * + ;
    - C (13)
      * lambda

}

* + - * + P (2)

F (29)

function

id

H (30)

T (9)

boolean

(

A (32)

T (7)

int

id

K (34)

,

T (7)

* int

id

K (34)

* ,
* T (7)
  + int
* id
* K (35)
  + lambda

)

{

C (12)

B (6)

S (19)

* return
* X (23)
  + E (36)
    - R (39)
      * U (43)
        + V (47)

id

D (52)

lambda

* + - * + U\_ (46)

lambda

* + - * R\_ (40)
        + <
        + U (43)

V (49)

entero

U\_ (46)

lambda

* + - * + R\_ (42)

lambda

* + - E\_ (37)
      * &&
      * R (39)
        + U (43)

V (47)

id

D (52)

lambda

U\_ (46)

lambda

* + - * + R\_ (41)

>

U (43)

V (49)

entero

U\_ (46)

lambda

R\_ (42)

lambda

* + - * E\_ (37)
        + &&
        + R (39)

U (43)

V (47)

id

D (52)

lambda

U\_ (46)

lambda

R\_ (40)

<

U (43)

V (49)

entero

U\_ (46)

lambda

R\_ (42)

lambda

* + - * + E\_ (37)

&&

R (39)

U (43)

V (47)

id

D (52)

* lambda

U\_ (46)

lambda

R\_ (41)

>

U (43)

V (47)

* id
* D (51)
  + (
  + L (25)
    - E (36)
      * R (39)
        + U (43)

V (47)

id

D (52)

lambda

U\_ (46)

lambda

* + - * + R\_ (42)

lambda

* + - * E\_ (38)
        + lambda
    - Q (27)
      * ,
      * E (36)
        + R (39)

U (43)

V (47)

id

D (52)

lambda

U\_ (46)

lambda

R\_ (42)

lambda

* + - * + E\_ (38)

lambda

* + - * Q (28)
        + lambda
  + )

U\_ (46)

* lambda

R\_ (42)

lambda

E\_ (38)

lambda

* ;

C (13)

lambda

}

P (2)

F (29)

function

id

H (31)

lambda

(

A (33)

lambda

)

{

C (12)

B (5)

* if
* (
* E (36)
  + R (39)
    - U (43)
      * V (47)
        + id
        + D (51)

(

L (25)

E (36)

R (39)

U (43)

* V (49)
  + entero
* U\_ (46)
  + lambda

R\_ (42)

* lambda

E\_ (38)

lambda

Q (27)

,

E (36)

R (39)

* U (43)
  + V (49)
    - entero
  + U\_ (46)
    - lambda
* R\_ (42)
  + lambda

E\_ (38)

* lambda

Q (27)

,

E (36)

* R (39)
  + U (43)
    - V (49)
      * entero
    - U\_ (46)
      * lambda
  + R\_ (42)
    - lambda
* E\_ (38)
  + lambda

Q (28)

* lambda

)

* + - * U\_ (46)
        + lambda
    - R\_ (42)
      * lambda
  + E\_ (38)
    - lambda
* )
* G (10)
  + S (17)
    - print
    - (
    - E (36)
      * R (39)
        + U (43)

V (49)

entero

U\_ (46)

lambda

* + - * + R\_ (42)

lambda

* + - * E\_ (38)
        + lambda
    - )
    - ;

C (12)

* B (6)
  + S (19)
    - return
    - X (24)
      * lambda
    - ;
* C (13)
  + lambda

}

P (1)

B (4)

let

T (7)

* int

id

;

P (1)

B (6)

* S (16)
  + id
  + W (22)
    - (
    - L (26)
      * lambda
    - )
    - ;

P (3)

* lambda

***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:

* P (1)
  + B (4)
    - let
    - T (7)
      * int
    - id
    - ;
  + P (1)
    - B (4)
      * let
      * T (9)
        + boolean
      * id
      * ;
    - P (1)
      * B (4)
        + let
        + T (7)

int

* + - * + id
        + ;
      * P (1)
        + B (4)

let

T (8)

string

id

;

* + - * + P (1)

B (4)

let

T (7)

int

id

;

P (1)

B (4)

let

T (9)

boolean

id

;

P (2)

F (29)

function

id

H (30)

* T (7)
  + int

(

A (32)

* T (7)
  + int
* id
* K (34)
  + ,
  + T (9)
    - boolean
  + id
  + K (35)
    - lambda

)

{

C (12)

* B (6)
  + S (17)
    - print
    - (
    - E (36)
      * R (39)
        + U (43)

V (47)

id

D (52)

lambda

U\_ (46)

lambda

* + - * + R\_ (42)

lambda

* + - * E\_ (38)
        + lambda
    - )
    - ;
* C (12)
  + B (6)
    - S (16)
      * id
      * W (21)
        + =
        + E (36)

R (39)

U (43)

V (47)

id

D (52)

* lambda

U\_ (44)

+

V (47)

* id
* D (52)
  + lambda

U\_ (46)

* lambda

R\_ (42)

lambda

E\_ (38)

lambda

* + - * + ;
  + C (12)
    - B (6)
      * S (16)
        + id
        + W (21)

=

E (36)

R (39)

U (43)

V (47)

* id
* D (52)
  + lambda

U\_ (46)

* lambda

R\_ (42)

lambda

E\_ (37)

&&

R (39)

U (43)

* V (47)
  + id
  + D (52)
    - lambda
* U\_ (46)
  + lambda

R\_ (42)

* lambda

E\_ (38)

lambda

;

* + - C (12)
      * B (6)
        + S (19)

return

X (23)

E (36)

R (39)

U (43)

* V (48)
  + (
  + E (36)
    - R (39)
      * U (43)
        + V (49)

entero

* + - * + U\_ (46)

lambda

* + - * R\_ (42)
        + lambda
    - E\_ (38)
      * lambda
  + )
* U\_ (46)
  + lambda

R\_ (42)

* lambda

E\_ (38)

lambda

;

* + - * C (13)
        + lambda

}

P (2)

F (29)

* function
* id
* H (30)
  + T (9)
    - boolean
* (
* A (32)
  + T (7)
    - int
  + id
  + K (34)
    - ,
    - T (9)
      * boolean
    - id
    - K (35)
      * lambda
* )
* {
* C (12)
  + B (6)
    - S (18)
      * input
      * (
      * id
      * )
      * ;
  + C (12)
    - B (6)
      * S (17)
        + print
        + (
        + E (36)

R (39)

U (43)

V (48)

(

E (36)

* R (39)
  + U (43)
    - V (49)
      * entero
    - U\_ (44)
      * +
      * V (49)
        + entero
      * U\_ (44)
        + +
        + V (49)

entero

* + - * + U\_ (44)

+

V (48)

(

E (36)

R (39)

U (43)

* V (49)
  + entero
* U\_ (45)
  + -
  + V (47)
    - id
    - D (52)
      * lambda
  + U\_ (46)
    - lambda

R\_ (42)

* lambda

E\_ (38)

lambda

)

U\_ (46)

lambda

* + R\_ (42)
    - lambda
* E\_ (38)
  + lambda

)

U\_ (46)

lambda

R\_ (42)

lambda

E\_ (38)

lambda

* + - * + )
        + ;
    - C (12)
      * B (6)
        + S (19)

return

X (23)

E (36)

R (39)

U (43)

* V (48)
  + (
  + E (36)
    - R (39)
      * U (43)
        + V (47)

id

D (52)

lambda

* + - * + U\_ (46)

lambda

* + - * R\_ (42)
        + lambda
    - E\_ (37)
      * &&
      * R (39)
        + U (43)

V (47)

id

D (52)

lambda

U\_ (46)

lambda

* + - * + R\_ (42)

lambda

* + - * E\_ (37)
        + &&
        + R (39)

U (43)

V (47)

id

D (52)

lambda

U\_ (46)

lambda

R\_ (42)

lambda

* + - * + E\_ (38)

lambda

* + )
* U\_ (46)
  + lambda

R\_ (42)

* lambda

E\_ (38)

lambda

;

* + - * C (13)
        + lambda
* }

P (1)

* B (6)
  + S (16)
    - id
    - W (21)
      * =
      * E (36)
        + R (39)

U (43)

V (47)

id

D (52)

lambda

U\_ (44)

+

V (49)

entero

U\_ (45)

-

V (47)

* id
* D (52)
  + lambda

U\_ (44)

* +
* V (49)
  + entero
* U\_ (44)
  + +
  + V (48)
    - (
    - E (36)
      * R (39)
        + U (43)

V (49)

entero

U\_ (44)

+

V (47)

id

D (52)

lambda

U\_ (45)

-

V (49)

entero

U\_ (46)

lambda

* + - * + R\_ (42)

lambda

* + - * E\_ (38)
        + lambda
    - )
  + U\_ (46)
    - lambda

R\_ (42)

lambda

* + - * + E\_ (38)

lambda

* + - * ;
* P (1)
  + B (6)
    - S (17)
      * print
      * (
      * E (36)
        + R (39)

U (43)

V (47)

id

D (51)

(

L (25)

* E (36)
  + R (39)
    - U (43)
      * V (47)
        + id
        + D (52)

lambda

* + - * U\_ (46)
        + lambda
    - R\_ (42)
      * lambda
  + E\_ (38)
    - lambda
* Q (27)
  + ,
  + E (36)
    - R (39)
      * U (43)
        + V (47)

id

D (51)

(

L (25)

E (36)

R (39)

* U (43)
  + V (49)
    - entero
  + U\_ (46)
    - lambda
* R\_ (42)
  + lambda

E\_ (38)

* lambda

Q (27)

,

E (36)

* R (39)
  + U (43)
    - V (47)
      * id
      * D (52)
        + lambda
    - U\_ (46)
      * lambda
  + R\_ (42)
    - lambda
* E\_ (38)
  + lambda

Q (28)

* lambda

)

* + - * + U\_ (46)

lambda

* + - * R\_ (42)
        + lambda
    - E\_ (38)
      * lambda
  + Q (28)
    - lambda

)

U\_ (46)

lambda

R\_ (42)

lambda

* + - * + E\_ (38)

lambda

* + - * )
      * ;
  + P (3)
    - lambda

***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:

* P (1)
  + B (4)
    - let
    - T (8)
      * string
    - id
    - ;
  + P (2)
    - F (29)
      * function
      * id
      * H (31)
        + lambda
      * (
      * A (33)
        + lambda
      * )
      * {
      * C (12)
        + B (6)

S (17)

print

(

E (36)

R (39)

U (43)

* V (50)
  + cadena
* U\_ (46)
  + lambda

R\_ (42)

* lambda

E\_ (38)

lambda

)

;

* + - * + C (12)

B (6)

S (18)

input

(

id

)

;

C (13)

lambda

* + - * }
    - P (2)
      * F (29)
        + function
        + id
        + H (31)

lambda

* + - * + (
        + A (32)

T (8)

string

id

K (35)

lambda

* + - * + )
        + {
        + C (12)

B (6)

S (17)

print

(

E (36)

R (39)

* U (43)
  + V (50)
    - cadena
  + U\_ (46)
    - lambda
* R\_ (42)
  + lambda

E\_ (38)

* lambda

)

;

C (12)

B (6)

S (17)

print

(

E (36)

* R (39)
  + U (43)
    - V (47)
      * id
      * D (52)
        + lambda
    - U\_ (46)
      * lambda
  + R\_ (42)
    - lambda
* E\_ (38)
  + lambda

)

;

C (13)

lambda

* + - * + }
      * P (1)
        + B (6)

S (16)

id

W (22)

(

L (26)

lambda

)

;

* + - * + P (1)

B (6)

S (16)

id

W (22)

(

L (25)

* E (36)
  + R (39)
    - U (43)
      * V (47)
        + id
        + D (52)

lambda

* + - * U\_ (46)
        + lambda
    - R\_ (42)
      * lambda
  + E\_ (38)
    - lambda
* Q (28)
  + lambda

)

;

P (3)

lambda

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

let int a ;

let int b ;

function operacion

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

{

return number;

}

Error sintáctico: se encuentra el token kAbierta cuando debería aparecer el token pCerrado

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

let int ; //Error en esta línea

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 sintáctico: se encuentra el token puntComa cuando debería 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 línea

x =

x + 6

+ z

+ 1

+ (2

- y

- 7);

Error sintáctico: se encuentra el token id cuando debería aparecer el token pCerrado