# Anexo Ejemplos Grupo 33

**Formado por:**

+ **Pedro Amaya Moreno**
+ **Juan López González**
+ **Pablo Rodríguez Beceiro**

## Ejemplo 1:

```
/* Programa de ejemplo con errores
Prueba errores lexicos
*/
var = 35;
function foo int (int n, int m) {
    m = m*a_23;
    m /= n*5 + 657;
    if(n > m & n < m+356) print "n is higher";
    return n*m;
}

let a string = "hola mundo"
/*valor supera max int*/
let i = 32768;

function bar string () {
    print a;
    /* error mas de 64 caracteres*/
    let _hola_2 = "0123456789 0123456789 0123456789 0123456789 0123456789 0123456789";
    print (_fola_2);
    return _hola_2;
}

function fibo int (int n) {
    if(n < 1) return 0;
    if(n < 2) return 1;
    return fibo(n - 1) + fibo(n - 2);
}
/*definicion funcion main

****
*/
function main() {
    print foo(2 , 3); print "\n"; print fibo(35);
    bar();
    a / b;
    let x;
    input x;
    if(x > 10 && x < 20) print(fibo(x));
    if(x < 5 | x > 24) print fibor(x + 10);
    return;
}
main();
/* fin
```

## Salida errores:

```
Error Lexico 42:
   Esperando '&', ha recibido ' ', en la linea 8
   -> Manejando el error lexico: Continuara como si hubiera leido un '&&'

Error Sintactico 61: Valor inesperado 'let', en la linea 14
    -> Pausando el semantico y continuando el lexico hasta encontrar punto de seguro de salto:
Error Lexico 47:
   El valor = 32768 supera el valor maximo de 32767, en la linea 14
   -> Manejando el error lexico: Continuara el sintactico como si hubiera leido un entero valido

+ Informacion del error sintactico anterior: Error en la declaracion del let, esperando let
Identificador Tipo [= Valor] ;
    -> Reiniciando analizador sintactico a partir de la linea 16 con la palabra 'function'

Error Sintactico 61: Valor inesperado '=', en la linea 19
    -> Pausando el semantico y continuando el lexico hasta encontrar punto de seguro de salto:
Error Lexico 48:
   En la cadena '0123456789 0123456789 0123456789 0123456789 0123456789 0123456789' se excede la
cantidad maxima de 64 caracteres, en la linea 19
   -> Manejando el error lexico: Continuara el sintactico como si hubiera leido una cadena valida

+ Informacion del error sintactico anterior: Error en la declaracion del let, esperando let
Identificador Tipo [= Valor] ;
```

```
    -> Reiniciando analizador sintactico a partir de la linea 20 con la palabra 'print'

Error Lexico 40:
   Ha llegado el caracter '-' no reconocido por el procesador, en la linea 27
   -> Manejando el error lexico: Pidiendo otro token al lexico

Error Sintactico 61: Valor inesperado 'constante entera', en la linea 27
    -> Pausando el semantico y continuando el lexico hasta encontrar punto de seguro de salto:
Error Lexico 40:
   Ha llegado el caracter '-' no reconocido por el procesador, en la linea 27
   -> Manejando el error lexico: Pidiendo otro token al lexico

+ Informacion del error sintactico anterior: Error en el return, esperando return [Valor] ;
    -> Reiniciando analizador sintactico a partir de la linea 29 con la palabra '}'

Error Lexico 44:
   Esperando '=' o '*', ha recibido ' ', en la linea 36
   -> Manejando el error lexico: No se puede diferenciar entre un error de inicio de comentario
('/*' o al escribir un '/=', asi que continuara solo el Lexico

Error Lexico 43:
   Esperando '|', ha recibido ' ', en la linea 40
   -> Manejando el error lexico: Continuara como si hubiera leido un '||'

Error Lexico 46:
   Finalizacion de fichero inesperada dentro de un comentario, en la linea 44
   -> Manejando el error lexico: Pidiendo otro token al lexico

+ Fin de panic, recuperando el semantico y el sintactico para finalizar analisis

Error Sintactico 61: Valor inesperado 'EOF', en la linea 44
Se han detectado errores semanticos en el codigo:
   + Error Semantico 92:
      En la funcion 'bar' se espera que se devuelva cadena, pero esta tratando de devolver entero,
antes de la linea 24
```

## Ejemplo 2:

```
/*programa de ejemplo con errores 2*/
/*esta funcion calcula el factorial de n mediante un bucle*/
function factorial int(int n) {
   let i int;/*declaracion de variable*/
   let pn int;
   if(n == 0)
      return 1;
   pn = n;

   return n*factorial(n - 1);
}
/*esta funcion le resta a num los n primeros numeros naturales*/
function sumaNprimeros int(int n,int num){
   let res;/*declaracion de variable*/

   res=num;
   if(n < num)
      return n + sumaNprimeros(n, num);
   return res;
}
let x int = 32798;
function main() {
   let n int = 234;
   let m int = 432;
   let num = sumaNprimeros(n, m);
   input n;
   input m;
   let num2 = factorial(sumaNprimeros(n, m));
   num2 /= x;
   if(num < num2 & num2 > 0)
      print "mayor";
   let s string = "hola;
   print(s);
   return;

}
print main();
```

## Salida errores:

```
Error Sintactico 61: Valor inesperado '=', en la linea 6
    -> Pausando el semantico y continuando el lexico hasta encontrar punto de seguro de salto:
+ Informacion del error sintactico anterior: Error en la declaracion del if, esperando if (
ValorBool ) SentenciaSimple
```

```
    -> Reiniciando analizador sintactico a partir de la linea 8 con la palabra 'identificador'

Error Lexico 40:
   Ha llegado el caracter '-' no reconocido por el procesador, en la linea 10
   -> Manejando el error lexico: Pidiendo otro token al lexico

Error Sintactico 61: Valor inesperado 'constante entera', en la linea 10
    -> Pausando el semantico y continuando el lexico hasta encontrar punto de seguro de salto:
+ Informacion del error sintactico anterior: Error en el return, esperando return [Valor] ;
    -> Reiniciando analizador sintactico a partir de la linea 11 con la palabra '}'

Error Sintactico 61: Valor inesperado ';', en la linea 14
    -> Pausando el semantico y continuando el lexico hasta encontrar punto de seguro de salto:
+ Informacion del error sintactico anterior: Error en la declaracion del let, esperando let
Identificador Tipo [= Valor] ;
    -> Reiniciando analizador sintactico a partir de la linea 16 con la palabra 'identificador'

Error Lexico 47:
   El valor = 32798 supera el valor maximo de 32767, en la linea 21
   -> Manejando el error lexico: Continuara el sintactico como si hubiera leido un entero valido

Error Sintactico 61: Valor inesperado '=', en la linea 25
    -> Pausando el semantico y continuando el lexico hasta encontrar punto de seguro de salto:
+ Informacion del error sintactico anterior: Error en la declaracion del let, esperando let
Identificador Tipo [= Valor] ;
    -> Reiniciando analizador sintactico a partir de la linea 26 con la palabra 'input'

Error Sintactico 61: Valor inesperado '=', en la linea 28
    -> Pausando el semantico y continuando el lexico hasta encontrar punto de seguro de salto:
+ Informacion del error sintactico anterior: Error en la declaracion del let, esperando let
Identificador Tipo [= Valor] ;
    -> Reiniciando analizador sintactico a partir de la linea 29 con la palabra 'identificador'

Error Lexico 42:
   Esperando '&', ha recibido ' ', en la linea 30
   -> Manejando el error lexico: Continuara como si hubiera leido un '&&'

Error Lexico 49:
   La cadena 'hola;' termina inesperadamente con un salto de linea, en la linea 33
   -> Manejando el error lexico: Debido a la cadena faltandole las comillas de cierre, devolvemos
solo un ';'

Error Sintactico 61: Valor inesperado ';', en la linea 33
    -> Pausando el semantico y continuando el lexico hasta encontrar punto de seguro de salto:
+ Informacion del error sintactico anterior: Error en la declaracion del let, esperando let
Identificador Tipo [= Valor] ;
    -> Reiniciando analizador sintactico a partir de la linea 33 con la palabra 'print'

Se han detectado errores semanticos en el codigo:
   + Error Semantico 92:
      En la funcion 'factorial' se espera que se devuelva entero, pero esta tratando de devolver
vacio, antes de la linea 13
   + Error Semantico 100:
      El identificador 'main' es una funcion que no devuelve ningun valor y se esta tratando de
usarla para asignar un valor, antes de la linea 37
```

## Ejemplo 3:

```
/*con errores semanticos*/
function funcion int (int n) {
    if(n+4 < n < 10) return u;
    if(n -16 > 1) return a+b+c;
   return v < 10;
}
function divhastaMen int (int n,int num){
   /*declaracion de variable*/
   res=num;
   b=1;
   if (100 < num < n)
      b = 2;
   switch(b) {
      case 1:
         return n;
      case 2;
         n /= (n + 1);
         return n;
   }
   return res;
}
function main() {
   print restaNprimeros(35 ,4);
    let x;
```

```
    input x;
    x= 20;
    if(x > 10 && x < 20) print divhastaMen(x , x*20));
    if(x < 5 || x > 24) print divhastaMen(x + 10 , x+500);
   /*switch con 2 case y default*/
   let a
   let b
   let c;
   c= true;
   b=4;
   a=2;
   switch(a) {
       case 2:
          print(a);
          print funcion(b)
          b = n*345;
          b+4;
          break;
   case 39494:
          print(b)
      print funcion(a)
      a = n*459;
      a+349-3;
           break;
    default:
           c = true;
   }
    return;
}
main();
return;
break;
```

## Salida errores:

```
Error Lexico 40:
   Ha llegado el caracter '-' no reconocido por el procesador, en la linea 4
   -> Manejando el error lexico: Pidiendo otro token al lexico

Error Sintactico 61: Valor inesperado 'constante entera', en la linea 4
    -> Pausando el semantico y continuando el lexico hasta encontrar punto de seguro de salto:
+ Informacion del error sintactico anterior: Error en la declaracion del if, esperando if (
ValorBool ) SentenciaSimple
    -> Reiniciando analizador sintactico a partir de la linea 5 con la palabra 'return'

Error Sintactico 60: Error de equiparacion, esperando : ha recibido ;, en la linea 16
    -> Pausando el semantico y continuando el lexico hasta encontrar punto de seguro de salto:
+ Informacion del error sintactico anterior: Error en la declaracion de los bloques del switch,
esperando case Valor : [SentSwitch]
    -> Reiniciando analizador sintactico a partir de la linea 20 con la palabra 'return'

Error Sintactico 61: Valor inesperado ';', en la linea 24
    -> Pausando el semantico y continuando el lexico hasta encontrar punto de seguro de salto:
+ Informacion del error sintactico anterior: Error en la declaracion del let, esperando let
Identificador Tipo [= Valor] ;
    -> Reiniciando analizador sintactico a partir de la linea 25 con la palabra 'input'

Error Sintactico 60: Error de equiparacion, esperando ; ha recibido ), en la linea 27
    -> Pausando el semantico y continuando el lexico hasta encontrar punto de seguro de salto:
+ Informacion del error sintactico anterior: Error en la declaracion del if, esperando if (
ValorBool ) SentenciaSimple
    -> Reiniciando analizador sintactico a partir de la linea 28 con la palabra 'if'

Error Sintactico 61: Valor inesperado 'let', en la linea 31
    -> Pausando el semantico y continuando el lexico hasta encontrar punto de seguro de salto:
+ Informacion del error sintactico anterior: Error en la declaracion del let, esperando let
Identificador Tipo [= Valor] ;
    -> Reiniciando analizador sintactico a partir de la linea 33 con la palabra 'identificador'

Error Sintactico 61: Valor inesperado 'identificador', en la linea 40
    -> Pausando el semantico y continuando el lexico hasta encontrar punto de seguro de salto:
+ Informacion del error sintactico anterior: Error en el print, esperando print [Valor] ;
    -> Reiniciando analizador sintactico a partir de la linea 41 con la palabra 'identificador'

Error Sintactico 61: Valor inesperado '+', en la linea 41
    -> Pausando el semantico y continuando el lexico hasta encontrar punto de seguro de salto:
+ Informacion del error sintactico anterior: Error en la asignacion de la variable 'b'
(Identificador = Valor ;)
    -> Reiniciando analizador sintactico a partir de la linea 42 con la palabra 'break'

Error Lexico 47:
   El valor = 39494 supera el valor maximo de 32767, en la linea 43
   -> Manejando el error lexico: Continuara el sintactico como si hubiera leido un entero valido
```

```
Error Sintactico 61: Valor inesperado 'print', en la linea 45
    -> Pausando el semantico y continuando el lexico hasta encontrar punto de seguro de salto:
+ Informacion del error sintactico anterior: Error en el print, esperando print [Valor] ;
    -> Reiniciando analizador sintactico a partir de la linea 47 con la palabra 'identificador'

Error Sintactico 61: Valor inesperado '+', en la linea 47
    -> Pausando el semantico y continuando el lexico hasta encontrar punto de seguro de salto:
Error Lexico 40:
   Ha llegado el caracter '-' no reconocido por el procesador, en la linea 47
   -> Manejando el error lexico: Pidiendo otro token al lexico

+ Informacion del error sintactico anterior: Error en la asignacion de la variable 'a'
(Identificador = Valor ;)
    -> Reiniciando analizador sintactico a partir de la linea 48 con la palabra 'break'

Se han detectado errores semanticos en el codigo:
   + Error Semantico 96:
      Con el operando <, se ha tratado de hacer bool < entero, cuando solo se puede entero < entero,
antes de la linea 3
   + Error Semantico 91:
      returns inconsistentes en los bloques de la funcion o del switch, devolviendo entero y bool al
mismo tiempo, antes de la linea 6
   + Error Semantico 96:
      Con el operando <, se ha tratado de hacer bool < entero, cuando solo se puede entero < entero,
antes de la linea 11
   + Error Semantico 85:
      El identificador 'restaNprimeros' no es una funcion y se esta tratando como si lo fuera, antes
de la linea 23
   + Error Semantico 84:
      Tratando de asignar un valor de un tipo bool al identificador 'c' de tipo entero, antes de la
linea 33
   + Error Semantico 84:
      Tratando de asignar un valor de un tipo bool al identificador 'c' de tipo entero, antes de la
linea 50
   + Error Semantico 80:
      Ha tratado de usar un return fuera de una funcion, antes de la linea 56
   + Error Semantico 82:
      Ha tratado de usar un break fuera de un switch, antes de la linea 57
```

## Ejemplo 4:

```
/*Ejemplo Mal*/
function fibo int (int n,int num){
   if( a < d || a > d)
      print(str);
   a /= d;
   return a;
}
let m boolean = true;
function main() {
   let x int;
   /*error falta nombre identificador*/
   let string = "hola";
   input x;
   x = foo(x);
   print x + 232;
}
main();
function dividir int (int n,int num){
   let res ;/*declaracion de variable*/
   let b int;
   res=num;
   b=1;
   /*error falta ), da error con el identificador*/
   if (num < n
      b = ; /*no ve este error, esta saltando lineas*/
   switch(b) {
      case 1:
         return n;
      case 2;
         n /= (n + 1);
         return n;
   }
   return res;
}
/*comentario sin fin
```

## Salida errores:

```
Error Sintactico 60: Error de equiparacion, esperando identificador ha recibido string, en la linea
12
```

```
    -> Pausando el semantico y continuando el lexico hasta encontrar punto de seguro de salto:
+ Informacion del error sintactico anterior: Error en la declaracion del let, esperando let
Identificador Tipo [= Valor] ;
    -> Reiniciando analizador sintactico a partir de la linea 13 con la palabra 'input'

Error Sintactico 61: Valor inesperado ';', en la linea 19
    -> Pausando el semantico y continuando el lexico hasta encontrar punto de seguro de salto:
+ Informacion del error sintactico anterior: Error en la declaracion del let, esperando let
Identificador Tipo [= Valor] ;
    -> Reiniciando analizador sintactico a partir de la linea 20 con la palabra 'let'

Error Sintactico 61: Valor inesperado 'identificador', en la linea 25
    -> Pausando el semantico y continuando el lexico hasta encontrar punto de seguro de salto:
+ Informacion del error sintactico anterior: Error en la declaracion del if, esperando if (
ValorBool ) SentenciaSimple
    -> Reiniciando analizador sintactico a partir de la linea 26 con la palabra 'switch'

Error Sintactico 60: Error de equiparacion, esperando : ha recibido ;, en la linea 29
    -> Pausando el semantico y continuando el lexico hasta encontrar punto de seguro de salto:
+ Informacion del error sintactico anterior: Error en la declaracion de los bloques del switch,
esperando case Valor : [SentSwitch]
    -> Reiniciando analizador sintactico a partir de la linea 33 con la palabra 'return'

Error Lexico 46:
   Finalizacion de fichero inesperada dentro de un comentario, en la linea 35
   -> Manejando el error lexico: Pidiendo otro token al lexico

Se han detectado errores semanticos en el codigo:
   + Error Semantico 85:
      El identificador 'foo' no es una funcion y se esta tratando como si lo fuera, antes de la
linea 14
```

## Ejemplo 5:

```
/*Ejemplo mal con errores sintacticos y lexicos*/
/*error excede max*/
let const int = 123123412;
/*error excede 64 car*/
let pal string =
"tooooooooloooonnnnggggggggggggggglsfñalkjdfñalskjdflaksjflksdfñalskdjfñlaskdjfñalkdj";

function fibo int (int n,int num){
   let a = 23; /*error*/
   /*error lexico*/
   let b string = "halksef;
   let c boolean;
   input c;
   if(c | const < 1000) /*error ||*/
      print const;
   if(n > m & n < const) /*error &&*/
      return const*a;
}

let a int;
let b int;
a = fibo(const, 1000);

print a;
print (a + b*a);

/*error fallo de comentario, solo continua lexico (lanzara tambien un error el sintactico)*/
ada / 23423;
```

## Salida errores:

```
Error Lexico 47:
   El valor = 123123412 supera el valor maximo de 32767, en la linea 3
   -> Manejando el error lexico: Continuara el sintactico como si hubiera leido un entero valido

Error Lexico 48:
   En la cadena
'tooooooooloooonnnnggggggggggggggglsfÃ±alkjdfÃ±alskjdflaksjflksdfÃ±alskdjfÃ±laskdjfÃ±alkdj' se
excede la cantidad maxima de 64 caracteres, en la linea 5
   -> Manejando el error lexico: Continuara el sintactico como si hubiera leido una cadena valida

Error Sintactico 61: Valor inesperado '=', en la linea 8
    -> Pausando el semantico y continuando el lexico hasta encontrar punto de seguro de salto:
+ Informacion del error sintactico anterior: Error en la declaracion del let, esperando let
Identificador Tipo [= Valor] ;
    -> Reiniciando analizador sintactico a partir de la linea 10 con la palabra 'let'

Error Lexico 49:
```

```
   La cadena 'halksef;' termina inesperadamente con un salto de linea, en la linea 11
   -> Manejando el error lexico: Debido a la cadena faltandole las comillas de cierre, devolvemos
solo un ';'

Error Sintactico 61: Valor inesperado ';', en la linea 11
    -> Pausando el semantico y continuando el lexico hasta encontrar punto de seguro de salto:
+ Informacion del error sintactico anterior: Error en la declaracion del let, esperando let
Identificador Tipo [= Valor] ;
    -> Reiniciando analizador sintactico a partir de la linea 11 con la palabra 'let'

Error Lexico 43:
   Esperando '|', ha recibido ' ', en la linea 13
   -> Manejando el error lexico: Continuara como si hubiera leido un '||'

Error Lexico 42:
   Esperando '&', ha recibido ' ', en la linea 15
   -> Manejando el error lexico: Continuara como si hubiera leido un '&&'

Error Lexico 44:
   Esperando '=' o '*', ha recibido ' ', en la linea 27
   -> Manejando el error lexico: No se puede diferenciar entre un error de inicio de comentario
('/*' o al escribir un '/=', asi que continuara solo el Lexico

+ Fin de panic, recuperando el semantico y el sintactico para finalizar analisis

Error Sintactico 61: Valor inesperado 'EOF', en la linea 29
Se han detectado errores semanticos en el codigo:
   + Error Semantico 81:
      El identificador 'c' usado para el input no es del tipo cadena o entero, antes de la linea 13
```

**Ejemplo 6:**

```
/*Ejemplo analizador sintactico correcto*/
let b int = 10;
let g boolean = false;
function foo int(int a, string c, boolean d) {
    if(d && g)
        return a + b;
    a /= a*(b + b);
    if((a > b+a) || a < a*b)
        return 134;
}
foo(23, "hola", true);
```

**Tokens:**

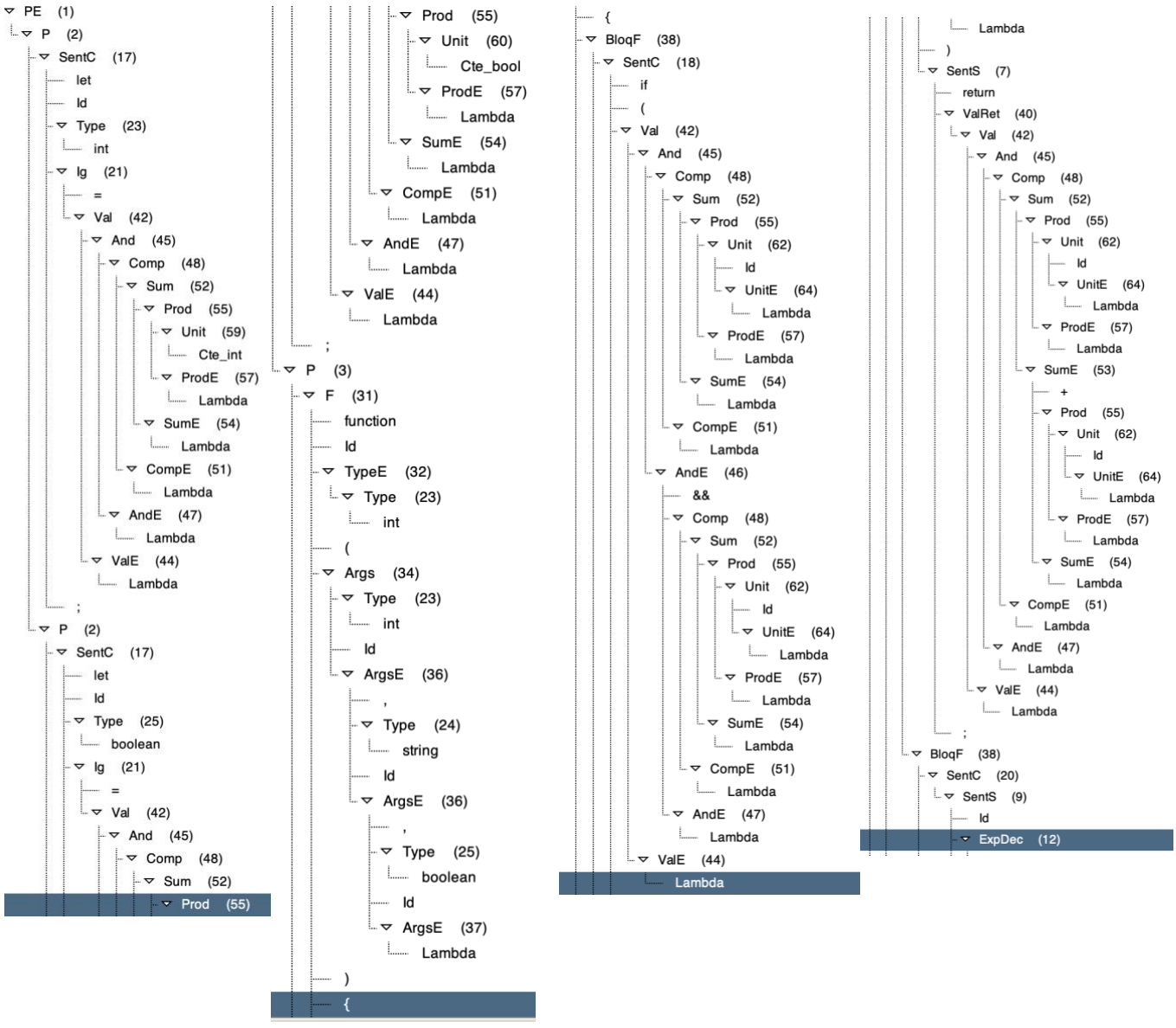
```
<19, >
<32, 1>
<20, >
<10, >
<1, 10>
<18, >
<19, >
<32, 2>
<22, >
<10, >
<3, 0>
<18, >
<25, >
<32, 3>
<20, >
<12, >
<20, >
<32, -1>
<17, >
<21, >
<32, -2>
<17, >
<22, >
<32, -3>
<13, >
<14, >
<27, >
<12, >
<32, -3>
<8, >
<32, 2>
<13, >
<26, >
<32, -1>
```

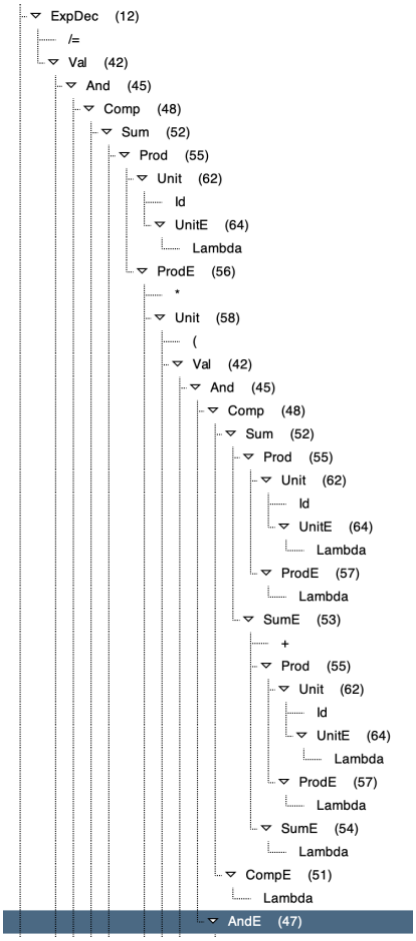
```
<4, >
<32, 1>
<18, >
<32, -1>
<11, >
<32, -1>
<5, >
<12, >
<32, 1>
<4, >
<32, 1>
<13, >
<18, >
<27, >
<12, >
<12, >
<32, -1>
<7, >
<32, 1>
<4, >
<32, -1>
<13, >
<9, >
<32, -1>
<6, >
<32, -1>
<5, >
<32, 1>
<13, >
<26, >
<1, 134>
<18, >
<15, >
<32, 3>
<12, >
<1, 23>
<17, >
<2, "hola">
<17, >
<3, 1>
<13, >
<18, >
<33, >
```

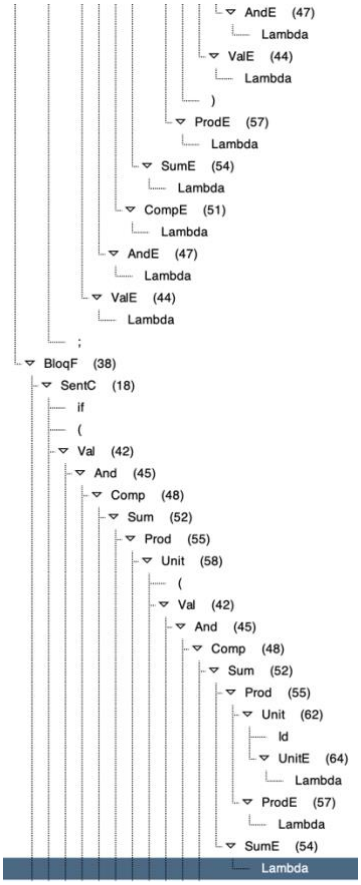
**Tabla Símbolos:**

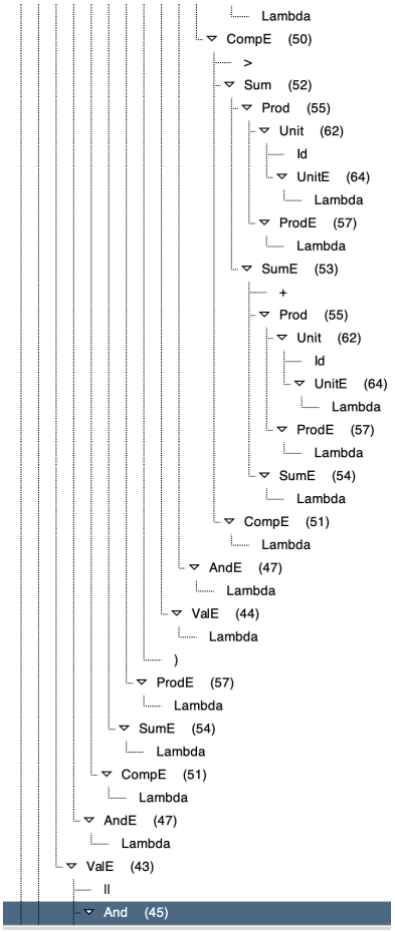
```
TABLA foo #2:
   * LEXEMA : 'a'
      + tipo : 'entero'
      + despl : 0
   * LEXEMA : 'c'
      + tipo : 'cadena'
      + despl : 2
   * LEXEMA : 'd'
      + tipo : 'bool'
      + despl : 130
TABLA Principal #1:
   * LEXEMA : 'b'
      + tipo : 'entero'
      + despl : 0
   * LEXEMA : 'g'
      + tipo : 'bool'
      + despl : 2
   * LEXEMA : 'foo'
      + tipo : 'funcion'
         + numParam :  3
         + TipoParam01 : 'entero'
         + TipoParam02 : 'cadena'
         + TipoParam03 : 'bool'
         + TipoRetorno : 'entero'
      + EtiqFuncion : 'Etfoo01'
Árbol sintáctico:
Descendente 1 2 17 23 21 42 45 48 52 55 59 57 54 51 47 44 2 17 25 21 42 45
48 52 55 60 57 54 51 47 44 3 31 32 23 34 23 36 24 36 25 37 38 18 42 45 48
```

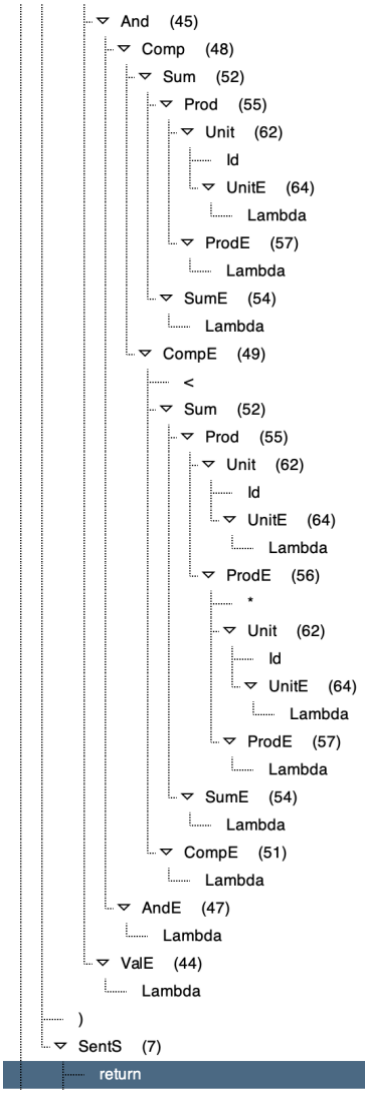
```
52 55 62 64 57 54 51 46 48 52 55 62 64 57 54 51 47 44 7 40 42 45 48 52 55
62 64 57 53 55 62 64 57 54 51 47 44 38 20 9 12 42 45 48 52 55 62 64 56 58
42 45 48 52 55 62 64 57 53 55 62 64 57 54 51 47 44 57 54 51 47 44 38 18 42
45 48 52 55 58 42 45 48 52 55 62 64 57 54 50 52 55 62 64 57 53 55 62 64 57
54 51 47 44 57 54 51 47 43 45 48 52 55 62 64 57 54 49 52 55 62 64 56 62 64
57 54 51 47 44 7 40 42 45 48 52 55 59 57 54 51 47 44 39 2 20 9 11 13 42 45
48 52 55 59 57 54 51 47 44 15 42 45 48 52 55 61 57 54 51 47 44 15 42 45 48
52 55 60 57 54 51 47 44 16 4
```

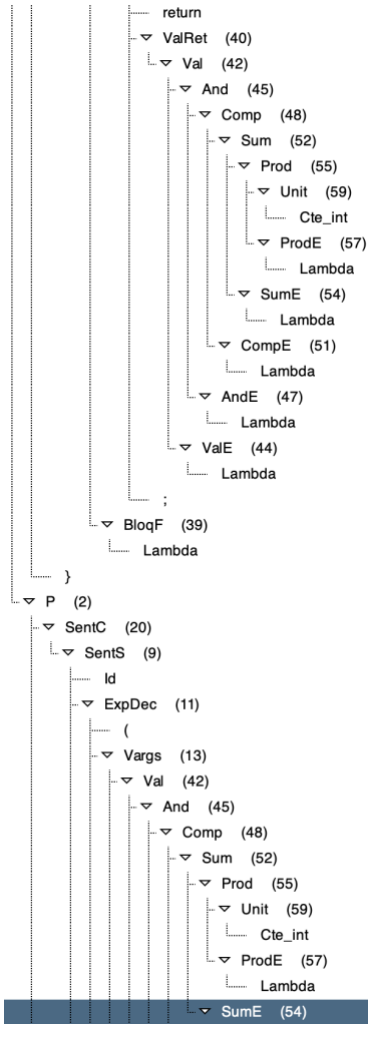


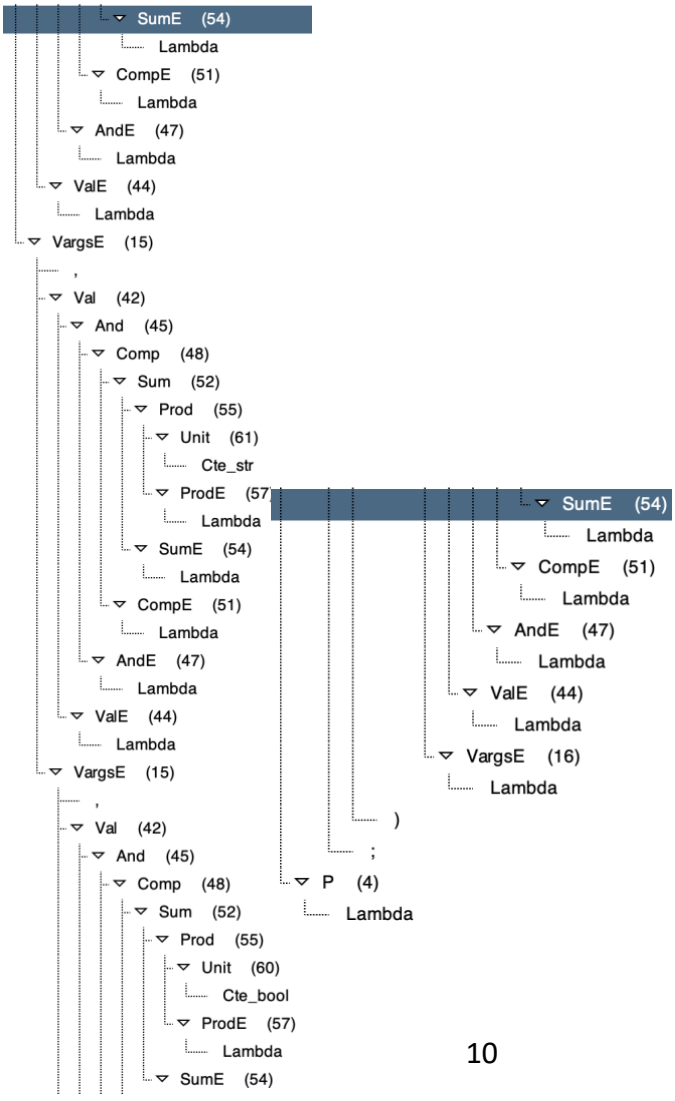

ExpDec (12)
/=
Val (42)
And (45)
Comp (48)
Sum (52)
Prod (55)
Unit (62)
Id
UnitE (64)
Lambda
ProdE (56)
*
Unit (58)
(
Val (42)
And (45)
Comp (48)
Sum (52)
Prod (55)
Unit (62)
Id
UnitE (64)
Lambda
ProdE (57)
Lambda
SumE (53)
+
Prod (55)
Unit (62)
Id
UnitE (64)
Lambda
ProdE (57)
Lambda
SumE (54)
Lambda
CompE (51)
Lambda
AndE (47)


AndE (47)
Lambda
ValE (44)
Lambda
)
ProdE (57)
Lambda
SumE (54)
Lambda
CompE (51)
Lambda
AndE (47)
Lambda
ValE (44)
Lambda
;
BloqF (38)
SentC (18)
if
(
Val (42)
And (45)
Comp (48)
Sum (52)
Prod (55)
Unit (58)
(
Val (42)
And (45)
Comp (48)
Sum (52)
Prod (55)
Unit (62)
Id
UnitE (64)
Lambda
ProdE (57)
Lambda
SumE (54)
Lambda


Lambda
CompE (50)
>
Sum (52)
Prod (55)
Unit (62)
Id
UnitE (64)
Lambda
ProdE (57)
Lambda
SumE (53)
+
Prod (55)
Unit (62)
Id
UnitE (64)
Lambda
ProdE (57)
Lambda
SumE (54)
Lambda
CompE (51)
Lambda
AndE (47)
Lambda
ValE (44)
Lambda
)
ProdE (57)
Lambda
SumE (54)
Lambda
CompE (51)
Lambda
AndE (47)
Lambda
ValE (43)
||
And (45)


And (45)
Comp (48)
Sum (52)
Prod (55)
Unit (62)
Id
UnitE (64)
Lambda
ProdE (57)
Lambda
SumE (54)
Lambda
CompE (49)
<
Sum (52)
Prod (55)
Unit (62)
Id
UnitE (64)
Lambda
ProdE (56)
*
Unit (62)
Id
UnitE (64)
Lambda
ProdE (57)
Lambda
SumE (54)
Lambda
CompE (51)
Lambda
AndE (47)
Lambda
ValE (44)
Lambda
)
SentS (7)
return


return
ValRet (40)
Val (42)
And (45)
Comp (48)
Sum (52)
Prod (55)
Unit (59)
Cte_int
ProdE (57)
Lambda
SumE (54)
Lambda
CompE (51)
Lambda
AndE (47)
Lambda
ValE (44)
Lambda
;
BloqF (39)
Lambda
}
P (2)
SentC (20)
SentS (9)
Id
ExpDec (11)
(
Vargs (13)
Val (42)
And (45)
Comp (48)
Sum (52)
Prod (55)
Unit (59)
Cte_int
ProdE (57)
Lambda
SumE (54)


SumE (54)
Lambda
CompE (51)
Lambda
AndE (47)
Lambda
ValE (44)
Lambda
VargsE (15)
,
Val (42)
And (45)
Comp (48)
Sum (52)
Prod (55)
Unit (61)
Cte_str
ProdE (57)
Lambda
SumE (54)
Lambda
CompE (51)
Lambda
AndE (47)
Lambda
ValE (44)
Lambda
VargsE (15)
,
Val (42)
And (45)
Comp (48)
Sum (52)
Prod (55)
Unit (60)
Cte_bool
ProdE (57)
Lambda
SumE (54)
)
;
P (4)
Lambda
SumE (54)
Lambda
CompE (51)
Lambda
AndE (47)
Lambda
ValE (44)
Lambda
VargsE (16)
Lambda

**Ejemplo 7:**

```
/*programa de ejemplo sin errores 2*/
/*esta funcion multiplica los n primeros numeros entre si*/
function divHastaCero int(int n) {
   print n;
   n /= 2;
   if(n < 1)
      return 0;
   return divHastaCero(n);
}
/*comprueba que n es mayor que 0 y menor que m*/
function mayor boolean (int n, int m){
   return (n > 0) && m >0 && n < m;
}
let x int;
let y int;
input x;
input y;
switch(x) {
   case 1:
      divHastaCero(y);
   case 2:
      if(mayor(y, x))
         y = y*x;
   default:
      print "caso no encontrado";
}
```

**Tokens:**

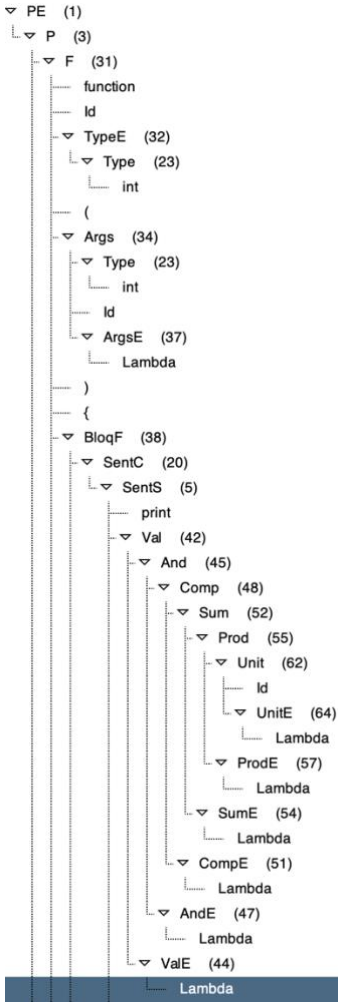
**<25, >**
**<32, 1>**
**<20, >**
**<12, >**
**<20, >**
**<32, -1>**
**<13, >**
**<14, >**
**<23, >**
**<32, -1>**
**<18, >**
**<32, -1>**
**<11, >**
**<1, 2>**
**<18, >**
**<27, >**
**<12, >**
**<32, -1>**
**<6, >**
**<1, 1>**
**<13, >**
**<26, >**
**<1, 0>**
**<18, >**
**<26, >**
**<32, 1>**
**<12, >**
**<32, -1>**
**<13, >**
**<18, >**
**<15, >**
**<25, >**
**<32, 2>**
**<22, >**
**<12, >**
**<20, >**
**<32, -1>**
**<17, >**
**<20, >**
**<32, -2>**
**<13, >**
**<14, >**
**<26, >**
**<12, >**
**<32, -1>**
**<7, >**
**<1, 0>**
**<13, >**
**<8, >**

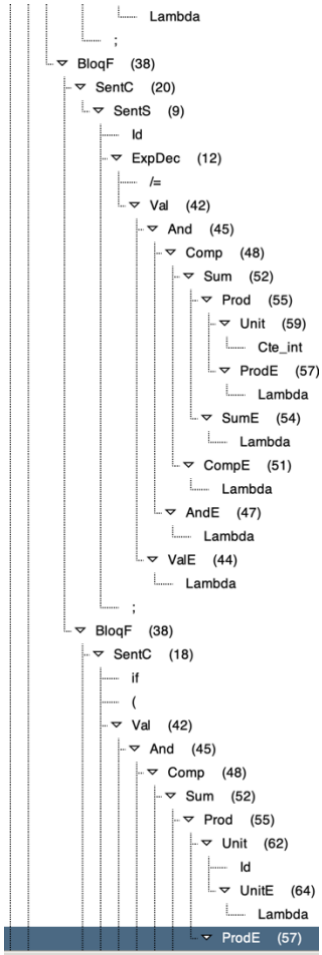
```
<32, -2>
<7, >
<1, 0>
<8, >
<32, -1>
<6, >
<32, -2>
<18, >
<15, >
<19, >
<32, 3>
<20, >
<18, >
<19, >
<32, 4>
<20, >
<18, >
<24, >
<32, 3>
<18, >
<24, >
<32, 4>
<18, >
<28, >
<12, >
<32, 3>
<13, >
<14, >
<29, >
<1, 1>
<16, >
<32, 1>
<12, >
<32, 4>
<13, >
<18, >
<29, >
<1, 2>
<16, >
<27, >
<12, >
<32, 2>
<12, >
<32, 4>
<17, >
<32, 3>
<13, >
<13, >
<32, 4>
<10, >
<32, 4>
<5, >
<32, 3>
<18, >
<30, >
<16, >
<23, >
<2, "caso no encontrado">
<18, >
<15, >
<33, >


Tabla Símbolos:
TABLA divHastaCero #2:
   * LEXEMA : 'n'
      + tipo : 'entero'
      + despl : 0
TABLA mayor #3:
   * LEXEMA : 'n'
      + tipo : 'entero'
      + despl : 0
   * LEXEMA : 'm'
      + tipo : 'entero'
      + despl : 2
TABLA Principal #1:
   * LEXEMA : 'divHastaCero'
      + tipo : 'funcion'
```

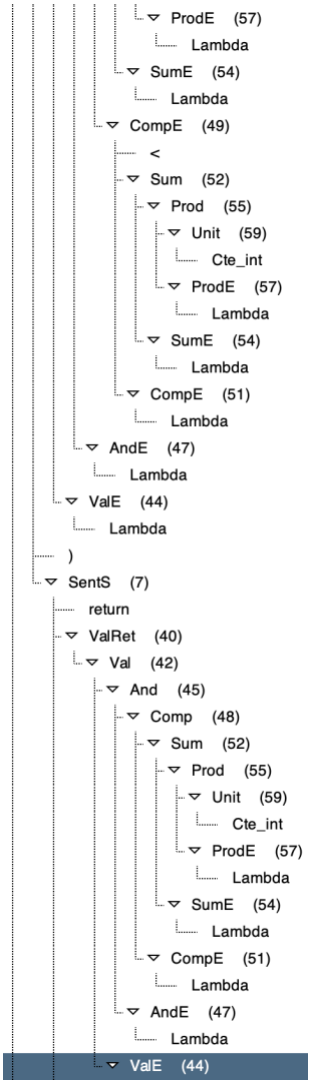
```
        + numParam :  1
        + TipoParam01 : 'entero'
        + TipoRetorno : 'entero'
     + EtiqFuncion : 'EtdivHastaCero01'
  * LEXEMA : 'mayor'
     + tipo : 'funcion'
        + numParam :  2
        + TipoParam01 : 'entero'
        + TipoParam02 : 'entero'
        + TipoRetorno : 'bool'
     + EtiqFuncion : 'Etmayor02'
  * LEXEMA : 'x'
     + tipo : 'entero'
     + despl : 0
  * LEXEMA : 'y'
     + tipo : 'entero'
     + despl : 2
```

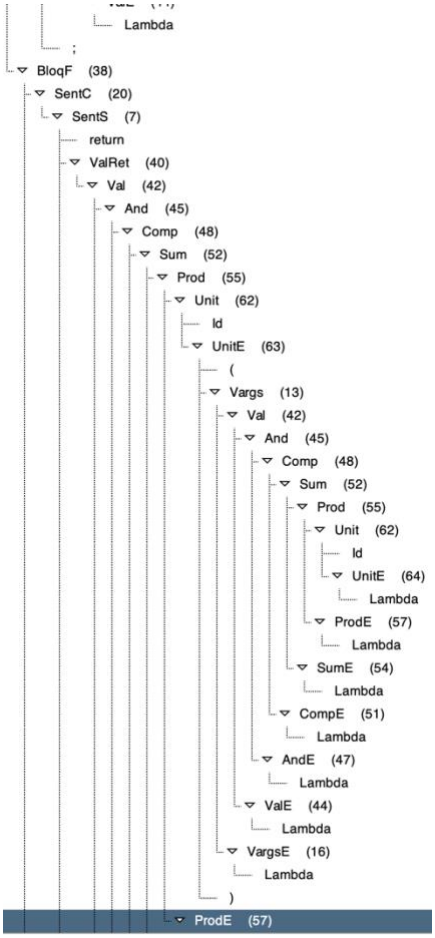
**Árbol sintáctico:**

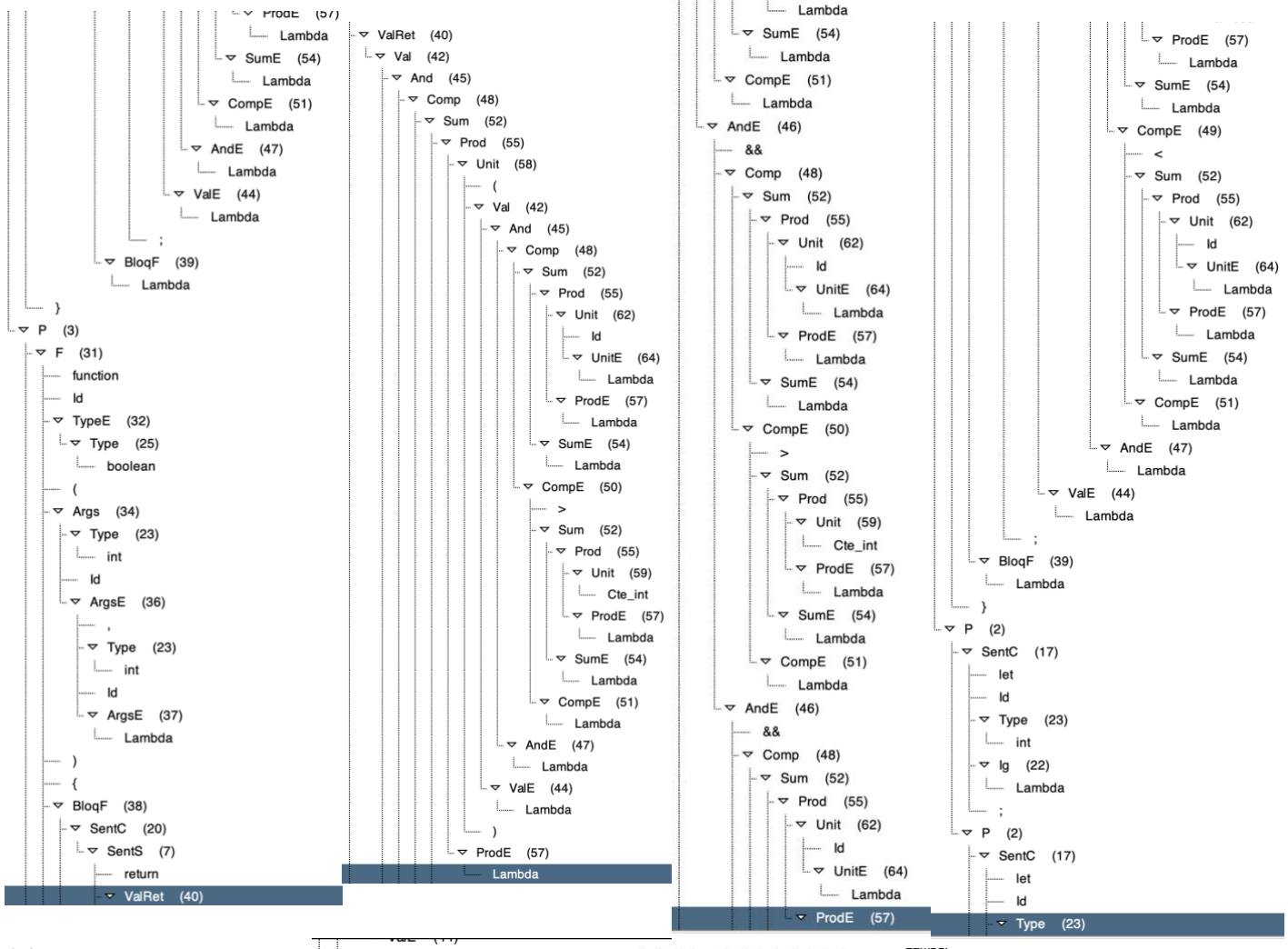
**Descendente** 1 3 31 32 23 34 23 37 38 20 5 42 45 48 52 55 62 64 57 54 51 47 44 38 20 9 12 42 45 48 52 55 59 57 54 51 47 44 38 18 42 45 48 52 55 62 64 57 54 49 52 55 59 57 54 51 47 44 7 40 42 45 48 52 55 59 57 54 51 47 44 38 20 7 40 42 45 48 52 55 62 63 13 42 45 48 52 55 62 64 57 54 51 47 44 16 57 54 51 47 44 39 3 31 32 25 34 23 36 23 37 38 20 7 40 42 45 48 52 55 58 42 45 48 52 55 62 64 57 54 50 52 55 59 57 54 51 47 44 57 54 51 46 48 52 55 62 64 57 54 50 52 55 59 57 54 51 46 48 52 55 62 64 57 54 49 52 55 62 64 57 54 51 47 44 39 2 17 23 22 2 17 23 22 2 20 6 2 20 6 2 19 42 45 48 52 55 62 64 57 54 51 47 44 26 42 45 48 52 55 59 57 54 51 47 44 29 20 9 11 13 42 45 48 52 55 62 64 57 54 51 47 44 16 30 26 42 45 48 52 55 59 57 54 51 47 44 29 18 42 45 48 52 55 62 63 13 42 45 48 52 55 62 64 57 54 51 47 44 15 42 45 48 52 55 62 64 57 54 51 47 44 16 57 54 51 47 44 9 10 42 45 48 52 55 62 64 56 62 64 57 54 51 47 44 30 27 29 20 5 42 45 48 52 55 61 57 54 51 47 44 30 4

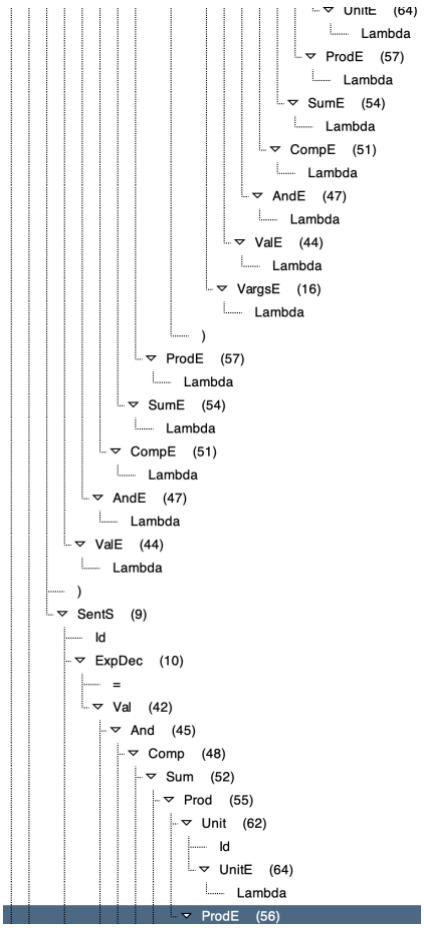


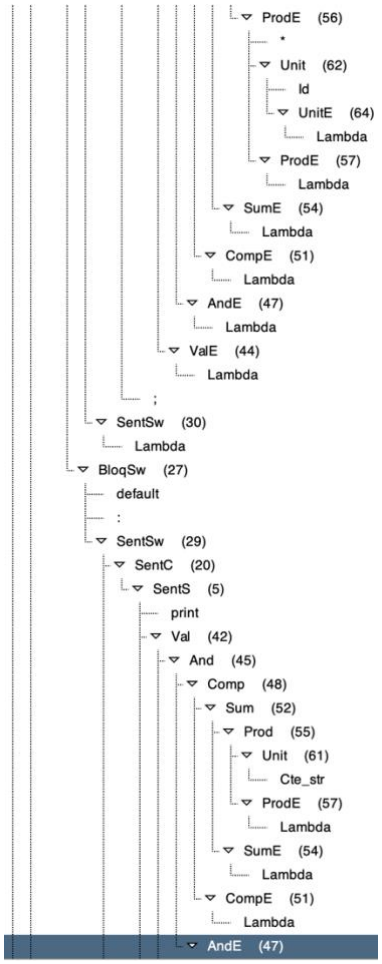



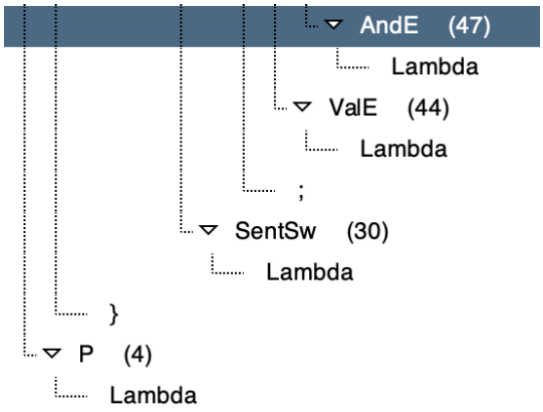




ProdE (57)
Lambda
SumE (54)
Lambda
CompE (51)
Lambda
AndE (47)
Lambda
ValE (44)
Lambda
;
BloqF (39)
Lambda
}
P (3)
F (31)
function
Id
TypeE (32)
Type (25)
boolean
(
Args (34)
Type (23)
int
Id
ArgsE (36)
,
Type (23)
int
Id
ArgsE (37)
Lambda
)
{
BloqF (38)
SentC (20)
SentS (7)
return
ValRet (40)
ValRet (40)
Val (42)
And (45)
Comp (48)
Sum (52)
Prod (55)
Unit (58)
(
Val (42)
And (45)
Comp (48)
Sum (52)
Prod (55)
Unit (62)
Id
UnitE (64)
Lambda
ProdE (57)
Lambda
SumE (54)
Lambda
CompE (50)
>
Sum (52)
Prod (55)
Unit (59)
Cte_int
ProdE (57)
Lambda
SumE (54)
Lambda
CompE (51)
Lambda
AndE (47)
Lambda
ValE (44)
Lambda
)
ProdE (57)
Lambda
Lambda
SumE (54)
Lambda
CompE (51)
Lambda
AndE (46)
&&
Comp (48)
Sum (52)
Prod (55)
Unit (62)
Id
UnitE (64)
Lambda
ProdE (57)
Lambda
SumE (54)
Lambda
CompE (50)
>
Sum (52)
Prod (55)
Unit (59)
Cte_int
ProdE (57)
Lambda
SumE (54)
Lambda
CompE (51)
Lambda
AndE (46)
&&
Comp (48)
Sum (52)
Prod (55)
Unit (62)
Id
UnitE (64)
Lambda
ProdE (57)
ProdE (57)
Lambda
SumE (54)
Lambda
CompE (49)
<
Sum (52)
Prod (55)
Unit (62)
Id
UnitE (64)
Lambda
ProdE (57)
Lambda
SumE (54)
Lambda
CompE (51)
Lambda
AndE (47)
Lambda
ValE (44)
Lambda
;
BloqF (39)
Lambda
}
P (2)
SentC (17)
let
Id
Type (23)
int
Ig (22)
Lambda
;
P (2)
SentC (17)
let
Id
Type (23)
Type (23)
int
Ig (22)
Lambda
;
P (2)
SentC (20)
SentS (6)
input
Id
;
P (2)
SentC (20)
SentS (6)
input
Id
;
P (2)
SentC (19)
switch
(
Val (42)
And (45)
Comp (48)
Sum (52)
Prod (55)
Unit (62)
Id
UnitE (64)
Lambda
ProdE (57)
Lambda
SumE (54)
Lambda
CompE (51)
Lambda
AndE (47)
Lambda
ValE (44)
Lambda
Lambda
)
{
BloqSw (26)
case
Val (42)
And (45)
Comp (48)
Sum (52)
Prod (55)
Unit (59)
Cte_int
ProdE (57)
Lambda
SumE (54)
Lambda
CompE (51)
Lambda
AndE (47)
Lambda
ValE (44)
Lambda
:
SentSw (29)
SentC (20)
SentS (9)
Id
ExpDec (11)
(
Vargs (13)
Val (42)
And (45)
Comp (48)
Sum (52)
Prod (55)
Unit (62)
Id
UnitE (64)
Lambda
ProdE (57)
Lambda
ProdE (57)
Lambda
SumE (54)
Lambda
CompE (51)
Lambda
AndE (47)
Lambda
ValE (44)
Lambda
VargsE (16)
Lambda
)
;
SentSw (30)
Lambda
BloqSw (26)
case
Val (42)
And (45)
Comp (48)
Sum (52)
Prod (55)
Unit (59)
Cte_int
ProdE (57)
Lambda
SumE (54)
Lambda
CompE (51)
Lambda
AndE (47)
Lambda
ValE (44)
Lambda
:
SentSw (29)
SentC (18)
if
(
(
Val (42)
And (45)
Comp (48)
Sum (52)
Prod (55)
Unit (62)
Id
UnitE (63)
(
Vargs (13)
Val (42)
And (45)
Comp (48)
Sum (52)
Prod (55)
Unit (62)
Id
UnitE (64)
Lambda
ProdE (57)
Lambda
SumE (54)
Lambda
CompE (51)
Lambda
AndE (47)
Lambda
ValE (44)
Lambda
VargsE (15)
,
Val (42)
And (45)
Comp (48)
Sum (52)
Prod (55)
Unit (62)
Id
UnitE (64)




**Ejemplo 8:**

```
/*Ejemplo Bien*/
/*funcion aleatoria*/
function main (int a, int b) {
    print a + b;
}
function aleatoria int (int n,int num){
   let _a_1 int;
   input _a_1;
   switch(_a_1){
      case 1:
         if(_a_1 < 10)
            print("hola mundo");
         break;
      case 2:
         _a_1 /= num;
      default:
         main(a , a*20);
   }
   return a;
}
print aleatoria(10, 50);
```

**Tokens:**

```
<25, >
<32, 1>
<12, >
<20, >
<32, -1>
<17, >
<20, >
<32, -2>
<13, >
<14, >
<23, >
<32, -1>
<4, >
<32, -2>
<18, >
<15, >
<25, >
<32, 2>
<20, >
<12, >
```

```
<20, >
<32, -1>
<17, >
<20, >
<32, -2>
<13, >
<14, >
<19, >
<32, -3>
<20, >
<18, >
<24, >
<32, -3>
<18, >
<28, >
<12, >
<32, -3>
<13, >
<14, >
<29, >
<1, 1>
<16, >
<27, >
<12, >
<32, -3>
<6, >
<1, 10>
<13, >
<23, >
<12, >
<2, "hola mundo">
<13, >
<18, >
<31, >
<18, >
<29, >
<1, 2>
<16, >
<32, -3>
<11, >
<32, -2>
<18, >
<30, >
<16, >
<32, 1>
<12, >
<32, 3>
<17, >
<32, 3>
<5, >
<1, 20>
<13, >
<18, >
<15, >
<26, >
<32, 3>
<18, >
<15, >
<23, >
<32, 2>
<12, >
<1, 10>
<17, >
<1, 50>
<13, >
<18, >
<33, >
```

**Tabla Símbolos:**

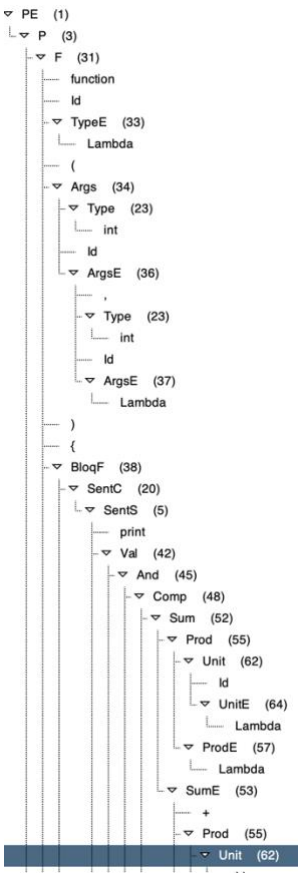
**TABLA main #2:**

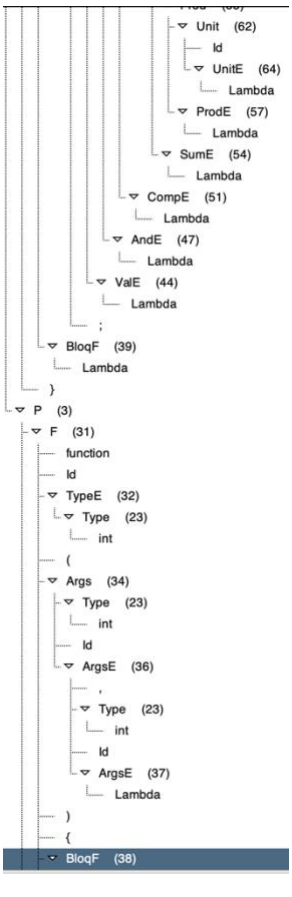
- **LEXEMA : 'a'**
  - **tipo : 'entero'**
  - **despl : 0**
- **LEXEMA : 'b'**
  - **tipo : 'entero'**
  - **despl : 2**

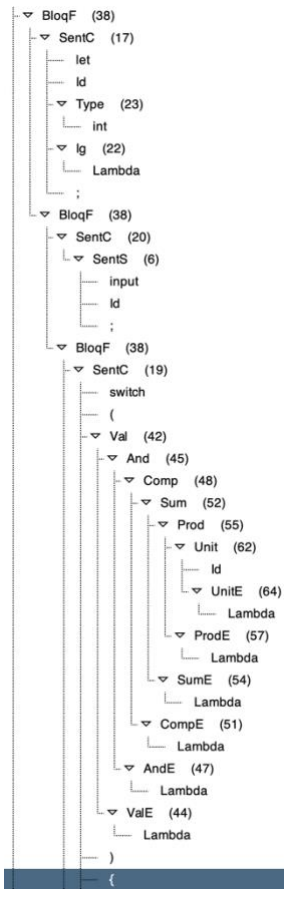
**TABLA aleatoria #3:**

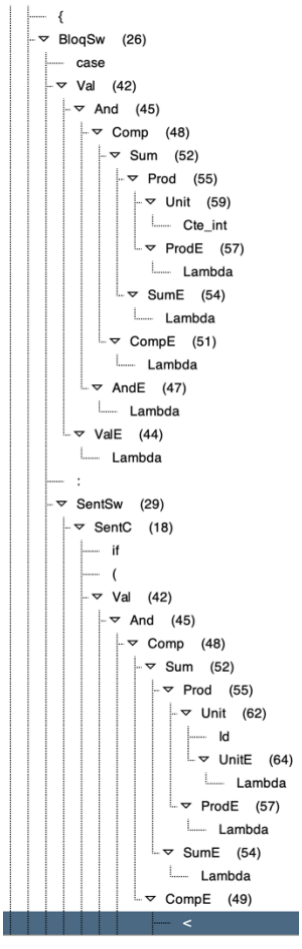
- **LEXEMA : 'n'**
  - **tipo : 'entero'**

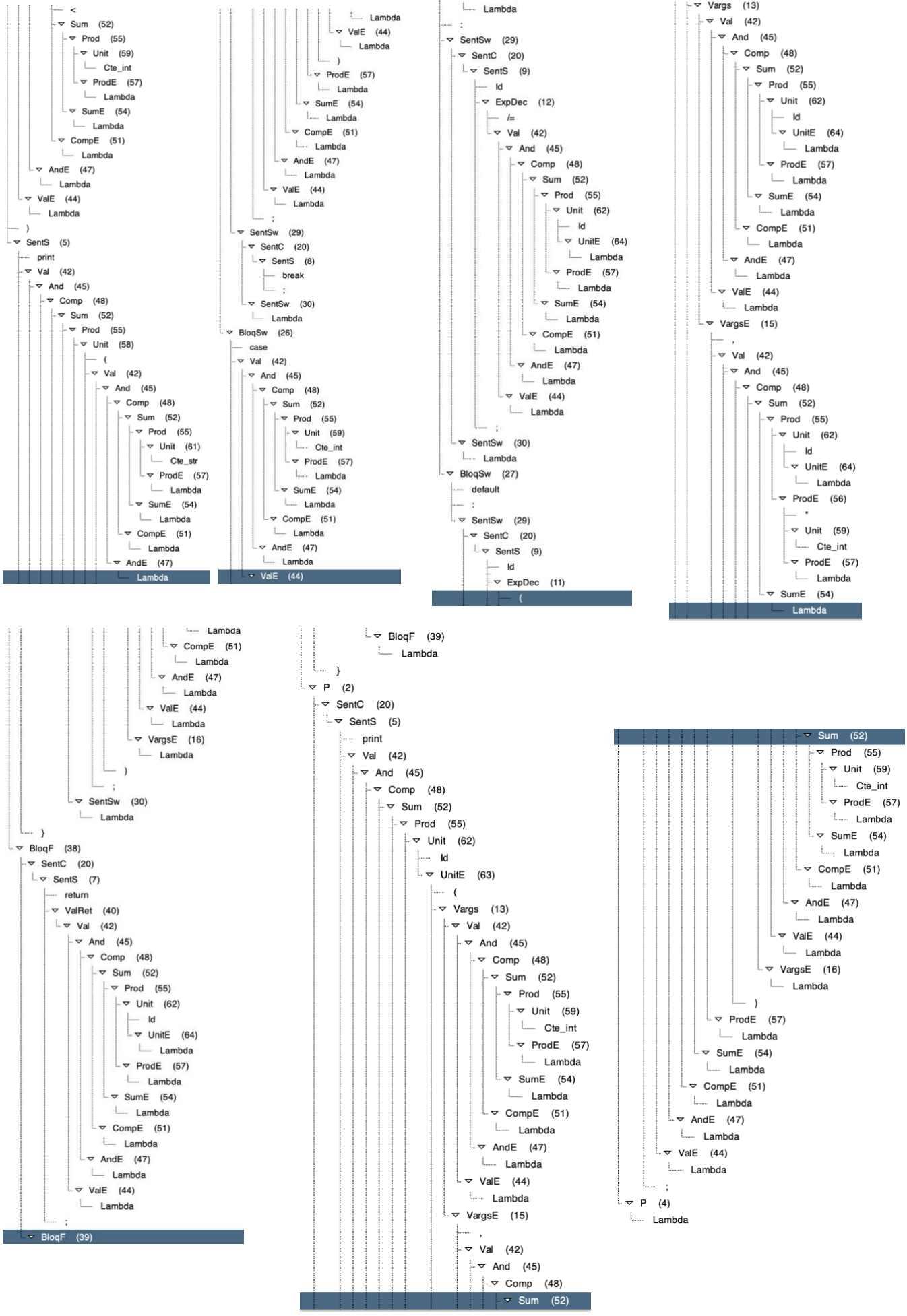
```
      + despl : 0
   * LEXEMA : 'num'
      + tipo : 'entero'
      + despl : 2
   * LEXEMA : '_a_1'
      + tipo : 'entero'
      + despl : 4
TABLA Principal #1:
   * LEXEMA : 'main'
      + tipo : 'funcion'
         + numParam :  2
         + TipoParam01 : 'entero'
         + TipoParam02 : 'entero'
         + TipoRetorno : 'vacio'
      + EtiqFuncion : 'Etmain01'
   * LEXEMA : 'aleatoria'
      + tipo : 'funcion'
         + numParam :  2
         + TipoParam01 : 'entero'
         + TipoParam02 : 'entero'
         + TipoRetorno : 'entero'
      + EtiqFuncion : 'Etaleatoria02'
   * LEXEMA : 'a'
      + tipo : 'entero'
      + despl : 0

Árbol sintáctico:
Descendente 1 3 31 33 34 23 36 23 37 38 20 5 42 45 48 52 55 62 64 57 53 55
62 64 57 54 51 47 44 39 3 31 32 23 34 23 36 23 37 38 17 23 22 38 20 6 38 19
42 45 48 52 55 62 64 57 54 51 47 44 26 42 45 48 52 55 59 57 54 51 47 44 29
18 42 45 48 52 55 62 64 57 54 49 52 55 59 57 54 51 47 44 5 42 45 48 52 55
58 42 45 48 52 55 61 57 54 51 47 44 57 54 51 47 44 29 20 8 30 26 42 45 48
52 55 59 57 54 51 47 44 29 20 9 12 42 45 48 52 55 62 64 57 54 51 47 44 30
27 29 20 9 11 13 42 45 48 52 55 62 64 57 54 51 47 44 15 42 45 48 52 55 62
64 56 59 57 54 51 47 44 16 30 38 20 7 40 42 45 48 52 55 62 64 57 54 51 47
44 39 2 20 5 42 45 48 52 55 62 63 13 42 45 48 52 55 59 57 54 51 47 44 15 42
45 48 52 55 59 57 54 51 47 44 16 57 54 51 47 44 4
```




< Sum (52) Prod (55) Unit (59) Cte_int ProdE (57) Lambda SumE (54) Lambda CompE (51) Lambda AndE (47) Lambda ValE (44) Lambda ) SentS (5) print Val (42) And (45) Comp (48) Sum (52) Prod (55) Unit (58) ( Val (42) And (45) Comp (48) Sum (52) Prod (55) Unit (61) Cte_str ProdE (57) Lambda SumE (54) Lambda CompE (51) Lambda AndE (47) Lambda
Lambda ValE (44) Lambda ) ProdE (57) Lambda SumE (54) Lambda CompE (51) Lambda AndE (47) Lambda ValE (44) Lambda ; SentSw (29) SentC (20) SentS (8) break ; SentSw (30) Lambda BloqSw (26) case Val (42) And (45) Comp (48) Sum (52) Prod (55) Unit (59) Cte_int ProdE (57) Lambda SumE (54) Lambda CompE (51) Lambda AndE (47) Lambda ValE (44)
ValE (44) Lambda : SentSw (29) SentC (20) SentS (9) Id ExpDec (12) /= Val (42) And (45) Comp (48) Sum (52) Prod (55) Unit (62) Id UnitE (64) Lambda ProdE (57) Lambda SumE (54) Lambda CompE (51) Lambda AndE (47) Lambda ValE (44) Lambda ; SentSw (30) Lambda BloqSw (27) default : SentSw (29) SentC (20) SentS (9) Id ExpDec (11) (
( Vargs (13) Val (42) And (45) Comp (48) Sum (52) Prod (55) Unit (62) Id UnitE (64) Lambda ProdE (57) Lambda SumE (54) Lambda CompE (51) Lambda AndE (47) Lambda ValE (44) Lambda VargsE (15) , Val (42) And (45) Comp (48) Sum (52) Prod (55) Unit (62) Id UnitE (64) Lambda ProdE (56) * Unit (59) Cte_int ProdE (57) Lambda SumE (54) Lambda
Lambda CompE (51) Lambda AndE (47) Lambda ValE (44) Lambda VargsE (16) Lambda ) ; SentSw (30) Lambda } BloqF (38) SentC (20) SentS (7) return ValRet (40) Val (42) And (45) Comp (48) Sum (52) Prod (55) Unit (62) Id UnitE (64) Lambda ProdE (57) Lambda SumE (54) Lambda CompE (51) Lambda AndE (47) Lambda ValE (44) Lambda ; BloqF (39)
BloqF (39) Lambda } P (2) SentC (20) SentS (5) print Val (42) And (45) Comp (48) Sum (52) Prod (55) Unit (62) Id UnitE (63) ( Vargs (13) Val (42) And (45) Comp (48) Sum (52) Prod (55) Unit (59) Cte_int ProdE (57) Lambda SumE (54) Lambda CompE (51) Lambda AndE (47) Lambda ValE (44) Lambda VargsE (15) , Val (42) And (45) Comp (48) Sum (52)
Sum (52) Prod (55) Unit (59) Cte_int ProdE (57) Lambda SumE (54) Lambda CompE (51) Lambda AndE (47) Lambda ValE (44) Lambda VargsE (16) Lambda ) ProdE (57) Lambda SumE (54) Lambda CompE (51) Lambda AndE (47) Lambda ValE (44) Lambda ; P (4) Lambda

**Ejemplo 9:**

```
let a int;
let b boolean;
let c string = "hola";
function imprimir(int n, string s) {
    print "Esta funcion imprime una cadena";
    if(n < 100)
        print("N menor que 100");
    print s;
}
input a;
imprimir(a, c);
a /= 10;
imprimir(a, "Tal vez se pueda ahora");
```

**Tokens:**

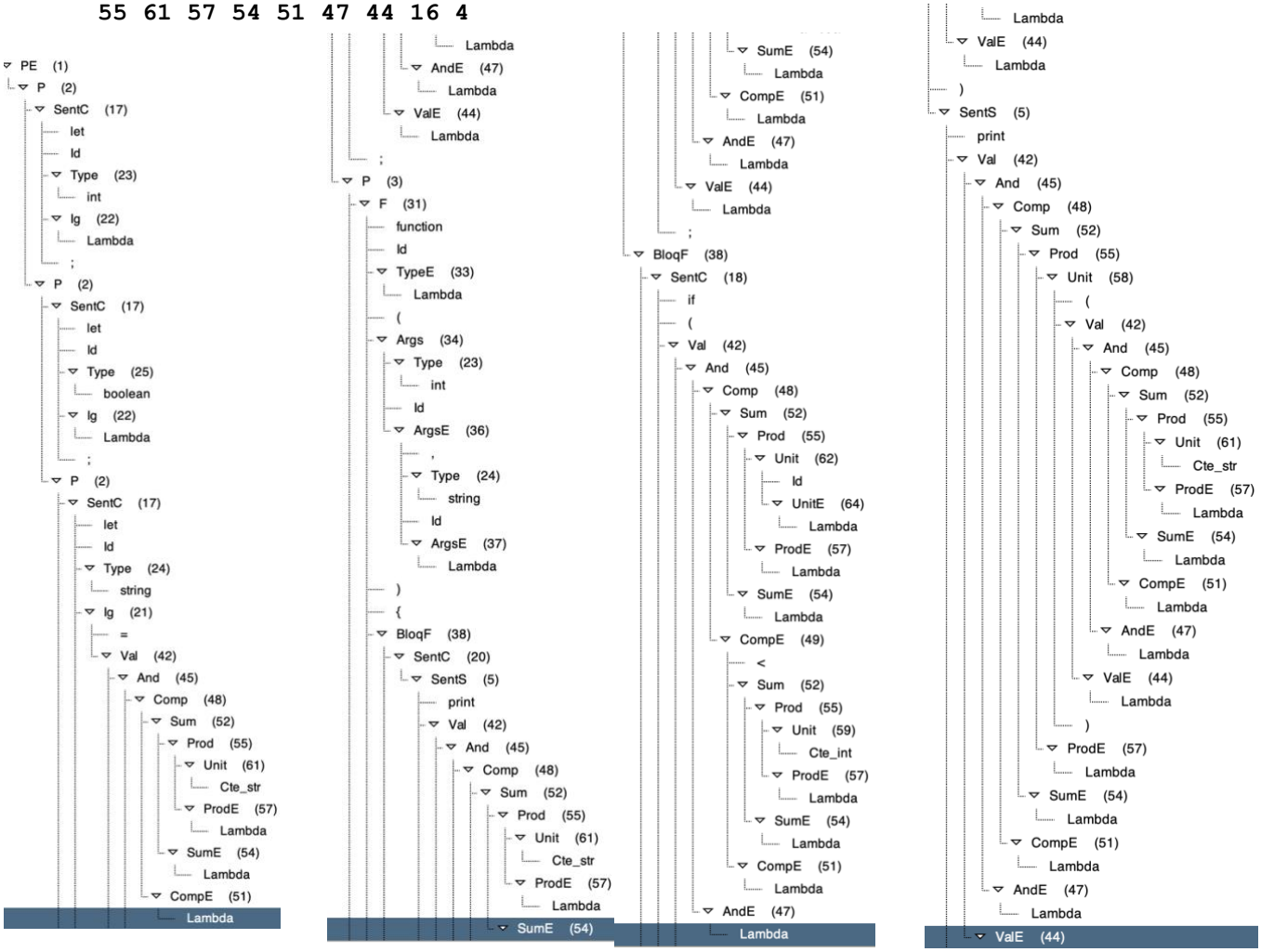
```
<19, >
<32, 1>
<20, >
<18, >
<19, >
<32, 2>
<22, >
<18, >
<19, >
<32, 3>
<21, >
<10, >
<2, "hola">
<18, >
<25, >
<32, 4>
<12, >
<20, >
<32, -1>
<17, >
<21, >
<32, -2>
<13, >
<14, >
<23, >
<2, "Esta funcion imprime una cadena">
<18, >
<27, >
<12, >
<32, -1>
<6, >
<1, 100>
<13, >
<23, >
<12, >
<2, "N menor que 100">
<13, >
<18, >
<23, >
<32, -2>
<18, >
<15, >
<24, >
<32, 1>
<18, >
<32, 4>
<12, >
<32, 1>
<17, >
<32, 3>
<13, >
<18, >
<32, 1>
<11, >
<1, 10>
<18, >
<32, 4>
<12, >
<32, 1>
<17, >
<2, "Tal vez se pueda ahora">
<13, >
<18, >
<33, >
```

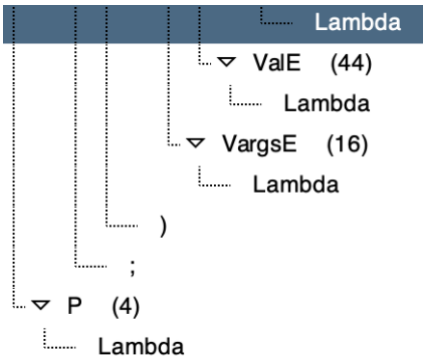
## Tabla Símbolos:

```
TABLA imprimir #2:
   * LEXEMA : 'n'
      + tipo : 'entero'
      + despl : 0
   * LEXEMA : 's'
      + tipo : 'cadena'
      + despl : 2
TABLA Principal #1:
   * LEXEMA : 'a'
      + tipo : 'entero'
      + despl : 0
   * LEXEMA : 'b'
      + tipo : 'bool'
      + despl : 2
   * LEXEMA : 'c'
      + tipo : 'cadena'
      + despl : 4
   * LEXEMA : 'imprimir'
      + tipo : 'funcion'
         + numParam :  2
         + TipoParam01 : 'entero'
         + TipoParam02 : 'cadena'
         + TipoRetorno : 'vacio'
      + EtiqFuncion : 'Etimprimir01'
```

## Árbol sintáctico:

**Descendente** 1 2 17 23 22 2 17 25 22 2 17 24 21 42 45 48 52 55 61 57 54 51 47 44 3 31 33 34 23 36 24 37 38 20 5 42 45 48 52 55 61 57 54 51 47 44 38 18 42 45 48 52 55 62 64 57 54 49 52 55 59 57 54 51 47 44 5 42 45 48 52 55 58 42 45 48 52 55 61 57 54 51 47 44 57 54 51 47 44 38 20 5 42 45 48 52 55 62 64 57 54 51 47 44 39 2 20 6 2 20 9 11 13 42 45 48 52 55 62 64 57 54 51 47 44 15 42 45 48 52 55 62 64 57 54 51 47 44 16 2 20 9 12 42 45 48 52 55 59 57 54 51 47 44 2 20 9 11 13 42 45 48 52 55 62 64 57 54 51 47 44 15 42 45 48 52 55 61 57 54 51 47 44 16 4




**Ejemplo 10:**

```
/*Ejemplo analizador sintactico correcto*/
function boo int(int n, int m) {
    if(n < m)
        return n*m;
    return n + m;
}
function foo string() {
    input x; input y;
    if(boo(x, y) < 100)
        return "Valores validos";
    return "Error";
}
input x;
switch(x) {
    case 1:
        foo();
        break;
    case 2:
        boo(100, x);
    default:
        print("No hemos hecho nada");
}
```

**Tokens:**

```
<25, >
<32, 1>
<20, >
<12, >
<20, >
<32, -1>
<17, >
<20, >
<32, -2>
<13, >
<14, >
<27, >
<12, >
<32, -1>
<6, >
<32, -2>
<13, >
<26, >
<32, -1>
<5, >
<32, -2>
<18, >
<26, >
<32, -1>
<4, >
<32, -2>
<18, >
<15, >
<25, >
<32, 2>
<21, >
<12, >
<13, >
<14, >
<24, >
<32, 3>
<18, >
<24, >
<32, 4>
<18, >
<27, >
<12, >
<32, 1>
<12, >
<32, 3>
<17, >
<32, 4>
<13, >
<6, >
<1, 100>
<13, >
<26, >
<2, "Valores validos">
<18, >
<26, >
<2, "Error">
<18, >
<15, >
<24, >
<32, 3>
<18, >
<28, >
<12, >
<32, 3>
<13, >
<14, >
<29, >
<1, 1>
<16, >
<32, 2>
<12, >
<13, >
<18, >
<31, >
<18, >
<29, >
<1, 2>
<16, >
<32, 1>
<12, >
```

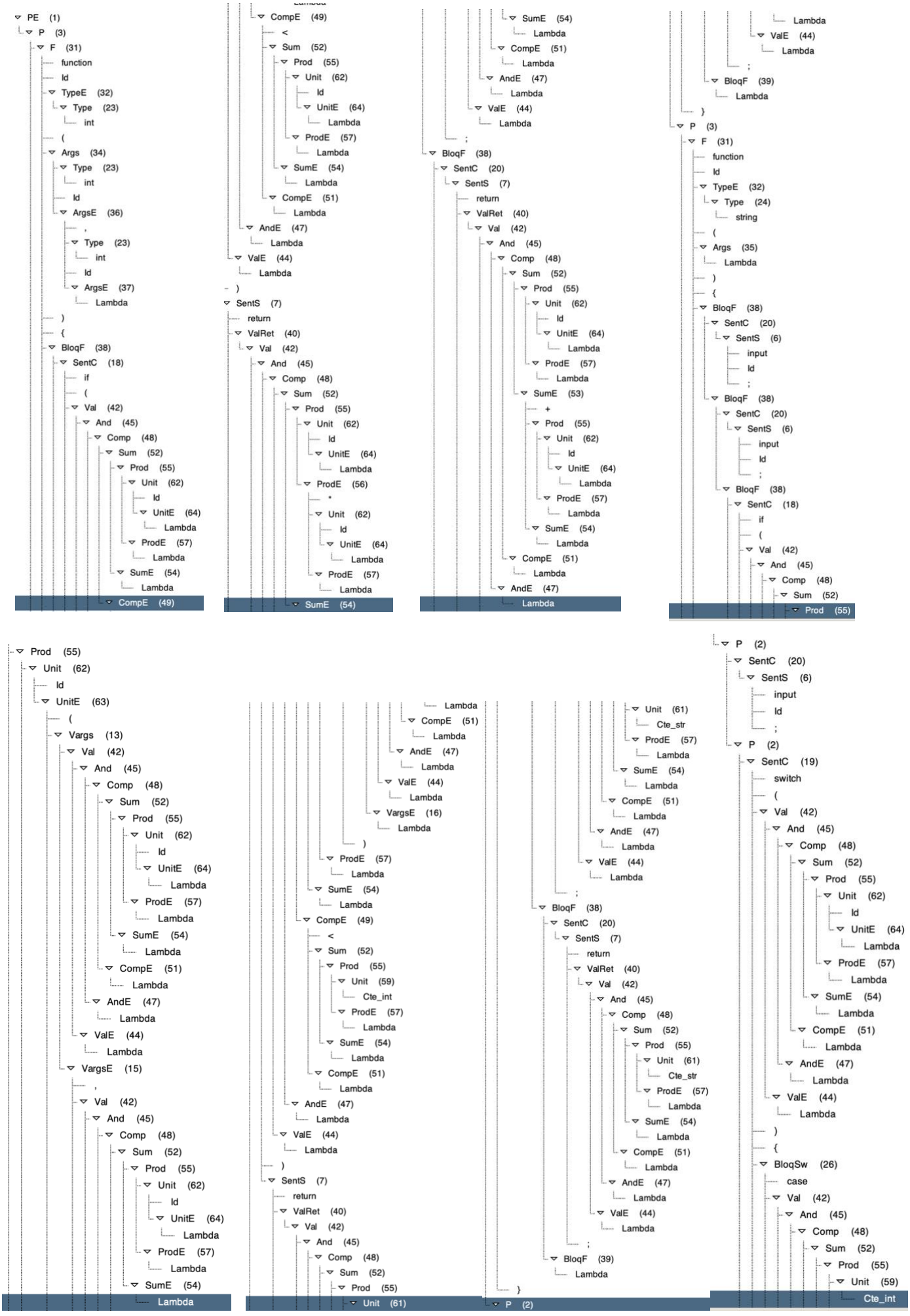
```
<1, 100>
<17, >
<32, 3>
<13, >
<18, >
<30, >
<16, >
<23, >
<12, >
<2, "No hemos hecho nada">
<13, >
<18, >
<15, >
<33, >
```

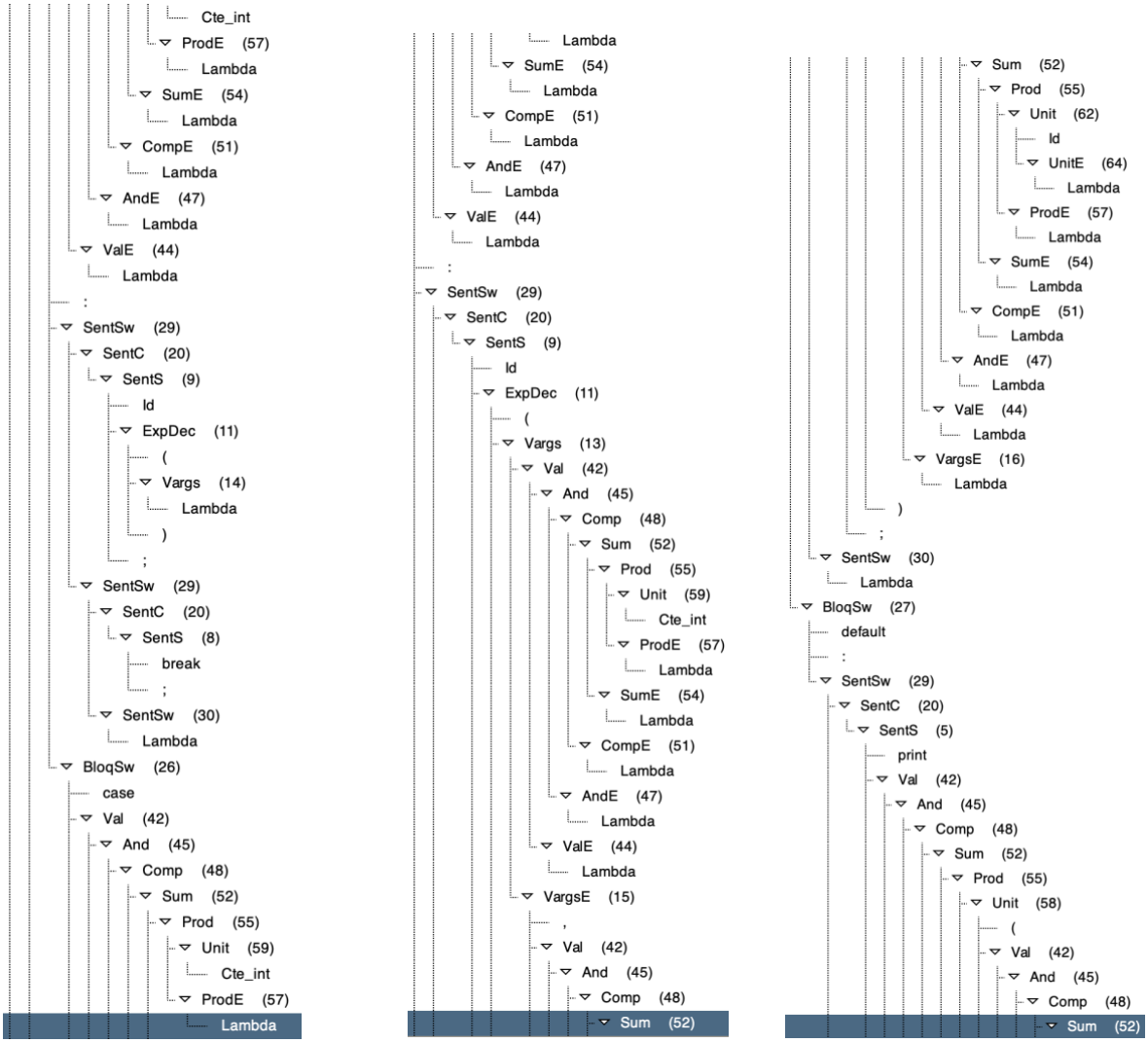
**Tabla Símbolos:**

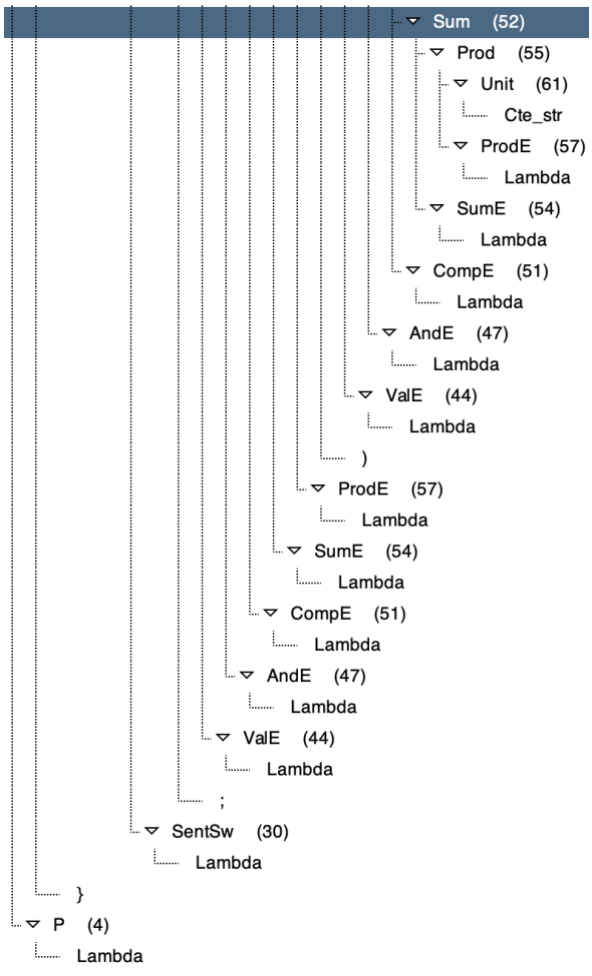
```
TABLA boo #2:
   * LEXEMA : 'n'
      + tipo : 'entero'
      + despl : 0
   * LEXEMA : 'm'
      + tipo : 'entero'
      + despl : 2
TABLA foo #3:
TABLA Principal #1:
   * LEXEMA : 'boo'
      + tipo : 'funcion'
         + numParam :  2
         + TipoParam01 : 'entero'
         + TipoParam02 : 'entero'
         + TipoRetorno : 'entero'
      + EtiqFuncion : 'Etboo01'
   * LEXEMA : 'foo'
      + tipo : 'funcion'
         + numParam :  1
         + TipoParam01 : 'vacio'
         + TipoRetorno : 'cadena'
      + EtiqFuncion : 'Etfoo02'
   * LEXEMA : 'x'
      + tipo : 'entero'
      + despl : 0
   * LEXEMA : 'y'
      + tipo : 'entero'
      + despl : 2
```

**Árbol sintáctico:**

```
Descendente 1 3 31 32 23 34 23 36 23 37 38 18 42 45 48 52 55 62 64 57 54 49
52 55 62 64 57 54 51 47 44 7 40 42 45 48 52 55 62 64 56 62 64 57 54 51 47
44 38 20 7 40 42 45 48 52 55 62 64 57 53 55 62 64 57 54 51 47 44 39 3 31 32
24 35 38 20 6 38 20 6 38 18 42 45 48 52 55 62 63 13 42 45 48 52 55 62 64 57
54 51 47 44 15 42 45 48 52 55 62 64 57 54 51 47 44 16 57 54 49 52 55 59 57
54 51 47 44 7 40 42 45 48 52 55 61 57 54 51 47 44 38 20 7 40 42 45 48 52 55
61 57 54 51 47 44 39 2 20 6 2 19 42 45 48 52 55 62 64 57 54 51 47 44 26 42
45 48 52 55 59 57 54 51 47 44 29 20 9 11 14 29 20 8 30 26 42 45 48 52 55 59
57 54 51 47 44 29 20 9 11 13 42 45 48 52 55 59 57 54 51 47 44 15 42 45 48
52 55 62 64 57 54 51 47 44 16 30 27 29 20 5 42 45 48 52 55 58 42 45 48 52
55 61 57 54 51 47 44 57 54 51 47 44 30 4
```


PE (1)
P (3)
F (31)
function
Id
TypeE (32)
Type (23)
int
(
Args (34)
Type (23)
int
Id
ArgsE (36)
,
Type (23)
int
Id
ArgsE (37)
Lambda
)
{
BloqF (38)
SentC (18)
if
(
Val (42)
And (45)
Comp (48)
Sum (52)
Prod (55)
Unit (62)
Id
UnitE (64)
Lambda
ProdE (57)
Lambda
SumE (54)
Lambda
CompE (49)
<
Sum (52)
Prod (55)
Unit (62)
Id
UnitE (64)
Lambda
ProdE (57)
Lambda
SumE (54)
Lambda
CompE (51)
Lambda
AndE (47)
Lambda
ValE (44)
Lambda
)
SentS (7)
return
ValRet (40)
Val (42)
And (45)
Comp (48)
Sum (52)
Prod (55)
Unit (62)
Id
UnitE (64)
Lambda
ProdE (56)
*
Unit (62)
Id
UnitE (64)
Lambda
ProdE (57)
Lambda
SumE (54)
Lambda
CompE (51)
Lambda
AndE (47)
Lambda
ValE (44)
Lambda
;
BloqF (38)
SentC (20)
SentS (7)
return
ValRet (40)
Val (42)
And (45)
Comp (48)
Sum (52)
Prod (55)
Unit (62)
Id
UnitE (64)
Lambda
ProdE (57)
Lambda
SumE (53)
+
Prod (55)
Unit (62)
Id
UnitE (64)
Lambda
ProdE (57)
Lambda
SumE (54)
Lambda
CompE (51)
Lambda
AndE (47)
Lambda
}
P (3)
F (31)
function
Id
TypeE (32)
Type (24)
string
(
Args (35)
Lambda
)
{
BloqF (38)
SentC (20)
SentS (6)
input
Id
;
BloqF (38)
SentC (20)
SentS (6)
input
Id
;
BloqF (38)
SentC (18)
if
(
Val (42)
And (45)
Comp (48)
Sum (52)
Prod (55)
Unit (62)
Id
UnitE (63)
(
Vargs (13)
Val (42)
And (45)
Comp (48)
Sum (52)
Prod (55)
Unit (62)
Id
UnitE (64)
Lambda
ProdE (57)
Lambda
SumE (54)
Lambda
CompE (51)
Lambda
AndE (47)
Lambda
ValE (44)
Lambda
VargsE (15)
,
Val (42)
And (45)
Comp (48)
Sum (52)
Prod (55)
Unit (62)
Id
UnitE (64)
Lambda
ProdE (57)
Lambda
SumE (54)
Lambda
)
ProdE (57)
Lambda
SumE (54)
Lambda
CompE (49)
<
Sum (52)
Prod (55)
Unit (59)
Cte_int
ProdE (57)
Lambda
SumE (54)
Lambda
CompE (51)
Lambda
AndE (47)
Lambda
ValE (44)
Lambda
)
SentS (7)
return
ValRet (40)
Val (42)
And (45)
Comp (48)
Sum (52)
Prod (55)
Unit (61)
Cte_str
ProdE (57)
Lambda
SumE (54)
Lambda
CompE (51)
Lambda
AndE (47)
Lambda
ValE (44)
Lambda
;
BloqF (38)
SentC (20)
SentS (7)
return
ValRet (40)
Val (42)
And (45)
Comp (48)
Sum (52)
Prod (55)
Unit (61)
Cte_str
ProdE (57)
Lambda
SumE (54)
Lambda
CompE (51)
Lambda
AndE (47)
Lambda
ValE (44)
Lambda
;
BloqF (39)
Lambda
}
P (2)
P (2)
SentC (20)
SentS (6)
input
Id
;
P (2)
SentC (19)
switch
(
Val (42)
And (45)
Comp (48)
Sum (52)
Prod (55)
Unit (62)
Id
UnitE (64)
Lambda
ProdE (57)
Lambda
SumE (54)
Lambda
CompE (51)
Lambda
AndE (47)
Lambda
ValE (44)
Lambda
)
{
BloqSw (26)
case
Val (42)
And (45)
Comp (48)
Sum (52)
Prod (55)
Unit (59)
Cte_int


Cte_int
ProdE (57)
Lambda
SumE (54)
Lambda
CompE (51)
Lambda
AndE (47)
Lambda
ValE (44)
Lambda
:
SentSw (29)
SentC (20)
SentS (9)
Id
ExpDec (11)
(
Vargs (14)
Lambda
)
;
SentSw (29)
SentC (20)
SentS (8)
break
;
SentSw (30)
Lambda
BloqSw (26)
case
Val (42)
And (45)
Comp (48)
Sum (52)
Prod (55)
Unit (59)
Cte_int
ProdE (57)
Lambda
Lambda
SumE (54)
Lambda
CompE (51)
Lambda
AndE (47)
Lambda
ValE (44)
Lambda
:
SentSw (29)
SentC (20)
SentS (9)
Id
ExpDec (11)
(
Vargs (13)
Val (42)
And (45)
Comp (48)
Sum (52)
Prod (55)
Unit (59)
Cte_int
ProdE (57)
Lambda
SumE (54)
Lambda
CompE (51)
Lambda
AndE (47)
Lambda
ValE (44)
Lambda
VargsE (15)
,
Val (42)
And (45)
Comp (48)
Sum (52)
Sum (52)
Prod (55)
Unit (62)
Id
UnitE (64)
Lambda
ProdE (57)
Lambda
SumE (54)
Lambda
CompE (51)
Lambda
AndE (47)
Lambda
ValE (44)
Lambda
VargsE (16)
Lambda
)
;
SentSw (30)
Lambda
BloqSw (27)
default
:
SentSw (29)
SentC (20)
SentS (5)
print
Val (42)
And (45)
Comp (48)
Sum (52)
Prod (55)
Unit (58)
(
Val (42)
And (45)
Comp (48)
Sum (52)


Sum (52)
Prod (55)
Unit (61)
Cte_str
ProdE (57)
Lambda
SumE (54)
Lambda
CompE (51)
Lambda
AndE (47)
Lambda
ValE (44)
Lambda
)
ProdE (57)
Lambda
SumE (54)
Lambda
CompE (51)
Lambda
AndE (47)
Lambda
ValE (44)
Lambda
;
SentSw (30)
Lambda
}
P (4)
Lambda