PRÁCTICA 2 (PECL2)

Teniendo en cuenta la especificación del lenguaje propuesto en la PECL1, se pide adaptar el analizador léxico y programar un analizador sintáctico y semántico, escrito en JFLex & CUP, que reconozca el lenguaje descrito en la PECL1. La gramática propuesta no debe tener conflictos y debe ser lo más sintética y legible posible.

Se deben emplear nombres significativos en los nombres de terminales y no terminales en la gramática. Por ejemplo: para hacer referencia al terminal que representa el token END: denotarlo simplemente como "*E*" sería incorrecto, mientras que un nombre autoexplicativo tal como "*TK_END*" sería correcto y por tanto deseable.

El analizador debe ser capaz de detectar los errores léxicos[1] (PECL1) y los siguientes errores sintácticos y semánticos:

Sintácticos

1. Errores en la declaración de variables. Por ejemplo: `id6 BOOL (falta el símbolo ->);`
2. Errores en la declaración de funciones. Por ejemplo: `function $nombre2 a2`
3. Expresiones mal formadas. Por ejemplo: `4 + + 5;` ó `id=(6 + 6;`
4. Invocaciones incorrectas a funciones (por sintaxis, no por semántica): por ejemplo una llamada a una función que pide dos parámetros será correcta si la llamada cumple con la sintaxis de llamada, independientemente de si se le pasan dos argumentos, tres o ninguno, así como de si los tipos de los mismos coinciden con los esperados, pues esto es parte del análisis semántico.

Semánticos

5. Se identifique el uso de variables que no han sido declaradas.
6. Se detecten llamadas a funciones que no han sido previamente declaradas.
7. Se lleve a cabo la comprobación de tipos. Por ejemplo: no asignar un valor entero a un identificador que ha sido declarado como BOOL.
8. Se compruebe que el número y tipo de parámetros de una llamada a una función coincide con los definidos en la declaración de la función.

Cuando detecte un error, el analizador propuesto deberá mostrar un mensaje que indique la línea donde se ha producido el error y el tipo de error. Dicho mensaje deberá ser significativo y la práctica se evaluará al alza cuanto más significativos y útiles para el usuario sean dichos mensajes.

[1] Se asume que los errores léxicos son capturados y tratados por el analizador léxico, el cual no es parte de la evaluación de esta práctica

Al final del análisis, y en caso de haber encontrado errores, se deberá mostrar un informe con el número de errores que se han producido de cada tipo: léxicos, sintácticos y semánticos. Por el contrario, si el analizador hubiera terminado sin encontrar ningún error deberá mostrar un mensaje indicando que ha finalizado sin errores, incluyendo a su vez el número de funciones correctamente declaradas. Si hay alguna función declarada pero no invocada en el código ha de emitirse un aviso (`warning`) al final.

Además, el analizador debe ser capaz de recuperarse cuando encuentre un error y continuar analizando el resto del programa.

Así, para el siguiente fichero:

```
1.      PROGRAM
2.       VARDECL
3.          id,id2,id3   -> INT;
4.          id4,id5,id6 -> BOOL;
5.          cont, asd    -> int;
6.       END;
7.
8.       function $nombre1 a1 (bool), b1 (int)
9.         id3 = 4 + b1;
10.       return id3;
11.
12.       function $nombre2 a2
13.       return 4;
14.
15.       function $nombre3
16.       return true;
17.
18.     BEGIN
19.         //fichero de prueba
20.         id3 = (6 + 6) - $nombre1 (true, 5) ;
21.         id  = $nombre3;
22.         id4 = $nombre1(false,2);
23.         id7 = $nombre3;
24.     END;
```

Declaración incompleta

Error concordancia de tipos

Variable no declarada

El analizador mostraría la siguiente salida:

```
Error sintáctico en línea 13: token leído return, se esperaba otro token
Error semántico en línea 21: Asignación de tipos incorrecto
Error semántico en línea 23: La variable no ha sido declarada
-------------------------------------------------------------
Número total de errores semánticos: 2
Número total de errores sintácticos: 1
Número total de errores léxicos: 0
-------------------------------------------------------------
Warning: la función $nombre3 ha sido declarada pero no usada
```