Práctica 4 - Analizador Sintáctico Descendente Recursivo

Procesadores de Lenguajes - 4º curso - Grado en Ingeniería Informática

1 Objetivos

- Elaborar una gramática LL(1) para expresiones aritméticas con operandos enteros.
- Desarrollar un Analizador Sintáctico Descendente Recursivo para la gramática generada utilizando los elementos que devuelve un analizador léxico basado en JFlex.

2 Especificaciones del lenguaje

- El analizador ha de comprobar la sintaxis de expresiones correspondientes a operaciones aritméticas entre números enteros.
- Las operaciones soportadas son: suma, resta, producto y división.
- Las expresiones utilizan notación infija, respetando la precedencia de las operaciones.
- Se acepta la utilización de paréntesis en múltiples niveles.
- Una expresión finaliza con el carácter ';' (punto y coma).

  Ejemplos de expresiones correctas:

```
5*4+6;
0-3*(2-    4);
50*40/6*(3+   (1-2)    ) / (93*4-(2+8 * 5));
5
+ 4;
314;
```

2.1 Especificaciones léxicas

El análisis léxico se encargará de procesar archivos con los siguientes tokens válidos:

INTEGER Número entero sin signo

ADD Operador suma ('+')

SUB Operador resta ('-')

TIMES Operador producto ('*')

DIV Operador division entera ('/')

LEFT_BRACKET Paréntesis izquierdo ('(')

RIGHT_BRACKET Paréntesis derecho (')')

SEMICOLON Terminador de expresión (';')

El analizador léxico deberá encargarse, además, de eliminar todos los espacios en blanco del archivo de entrada y de detectar los errores de tipo léxico.

2.2 Especificación sintáctica

A continuación se propone una gramática para el lenguaje descrito:

$E \rightarrow E+T; \mid E-T; \mid T;$
$T \rightarrow T*F \mid T/F \mid F$
$F \rightarrow (E) \mid \mathbf{int}$

Una sentencia, S, es una expresión, E, seguida de un carácter punto y coma (';').

Una expresión, a su vez, se forma mediante otra expresión seguida del operador suma o resta y de un término. También puede consistir en un único término.

Un término, T, está compuesto por otro término conectado a un factor mediante los operadores producto o división. Un término puede constar de un único factor.

Un factor, F, puede ser una expresión entre paréntesis o, simplemente, un número entero, **int**.

3 Instrucciones

1. Construir una clase Java, **Yytoken**, con las siguientes características:

   - Su propiedades incluyen, al menos, los siguientes valores: lexema, tipo de token, línea y columna en que aparece el lexema en el archivo que se analiza. Los valores de estas propiedades se asignan en el constructor.
   - Tiene un método público, **errorMatch( )**, que recibe como entrada una cadena con un tipo de token. Si el tipo de token es diferente del tipo de token del objeto, entonces se muestra por pantalla el siguiente mensaje de error:

     ```
     [Parser] Encontrado token de tipo #### (línea ##, columna ##)
     Se esperaba un token de tipo ####
     ```

2. Construir un analizador léxico basado en JFlex que imprima los tokens encontrados y los errores detectados utilizando los siguientes formatos:

   ```
   [Lex] (TOKEN, lexema)
   [Lex] Error léxico en (línea ##, columna ##)
   ```

3. El analizador léxico devolverá objetos de la clase Yytoken, que serán utilizados por el analizador sintáctico.
   Notas:

   (a) Puesto que el analizador léxico va a ser utilizado por un analizador sintáctico, el archivo JFlex no llevará la opción **%standalone**.

   (b) Para que la función **yylex()** del analizador léxico devuelva objetos de tipo Yytoken, es necesario especificar la opción **%type Yytoken** en el archivo JFlex

4. Si es necesario, modificar la gramática propuesta para que sea adecuada a un analizador sintáctico descendente recursivo, comprobando que es LL(1):

   - Limpiar la gramática
   - Eliminar recursividades a izquierda
   - Factorizar
   - Calcular los símbolos directores para todos los no terminales
   - Comprobar que la gramática es LL(1)

5. Programar una analizador sintáctico descendente recursivo sin retroceso para la gramática LL(1) que utilice la información proporcionada por el analizador léxico anterior. No se utilizará ninguna herramienta para generar analizadores sintácticos.

   - Cada vez que se encuentre un token de tipo correcto según la gramática, se mostrará un mensaje con el siguiente formato:

     ```
     [Parser]  (tipodeToken, lexema): Ok
     ```

   - Recuperación ante errores de sintaxis. Cuando el analizador sintáctico encuentre un token que no se adecúe a la gramática, ejecutará el método **errorMatch()** del objeto Yytoken para mostrar un mensaje de error. A continuación seguirá analizando el archivo saltando todos los tokens hasta encontrar un punto y coma. A partir de esa posición se reanudará el análisis sintáctico

Se entregarán, al menos, los siguientes archivos:

1. Especificación léxica para JFlex (.jflex)
2. Clase Yytoken (.java)

3. Clase analizador léxico generada por JFlex (.java)
4. Clase analizador sintáctico (.java)
5. Un archivo de entrada