Traductores de Lenguajes de Programación

Evaluación contínua

Convocatoria de Junio-2018

Actividad Sintáctica

Objetivo

Se pretende desarrollar un analizador sintáctico para un lenguaje L haciendo uso del generador automático de programas `JavaCC`.

Descripción del lenguaje L

Es un lenguaje de formato libre que diferencia mayúsculas de minúsculas. Los comentarios, espacios en blanco, tabuladores y fines de línea hacen de separadores de piezas sintácticas. También hace de separador dentro de una definición de una pieza sintáctica cualquier carácter que no se ajuste a su definición lexicográfica.

Un programa sintácticamente válido en el lenguaje L es la definición secuencial ordenada de declaraciones de constantes, de variables y de definiciones de subprogramas que acaba en la definición de un cuerpo con sentencias ejecutables que finaliza en un '.'. El formato se detalla seguidamente:

Al principio se define la cabecera del programa compuesta por: la palabra reservada `PROGRAMA` seguida del identificador del programa y del finalizador ';'

La declaración de constantes es opcional. Una declaración de constantes comienza por la palabra reservada `CONSTANTES` que tiene que ir seguida de al menos una declaración de una constante. Una declaración de una constante se define escribiendo secuencialmente el identificador de la constante, el símbolo separador '=' y el valor asociado a la constante. Un valor asociado a una constante puede ser otro identificador de constante o bien a una constante anónima de los tipos entero, real, carácter, o literal cadena de caracteres, o bien alguno de los valores de las constantes lógicas representados por las palabras reservadas `TRUE` O `FALSE`. Toda declaración de una constante finaliza con el símbolo ';'.

La declaración de variables es opcional. Una declaración de variables comienza por la palabra reservada `VARIABLES` seguida de al menos una lista de declaración de variables. Una lista de declaración de variables se define escribiendo secuencialmente una lista de identificadores no vacía separados por el símbolo ',', el símbolo separador ':' y un identificador de tipo. Toda lista de declaración de variables termina con el finalizador ';'.

Una lista de identificadores puede incluir para un determinado identificador una inicialización. La inicialización de un identificador de variable comienza por símbolo de asignación definido por la secuencia ':=' seguido de una expresión constante, que se define como una expresión simple formada por un solo operando constante que puede ser una constante anónima de los tipos entero, real, carácter, o literal cadena de caracteres, o bien alguno de los valores lógicos `TRUE` O `FALSE`.

La declaración de subprogramas es opcional, si existe puede repetirse secuencialmente 0 o más veces.
Una declaración de un subprograma comienza por la definición de la cabecera de subprograma y le sigue el cuerpo del subprograma finalizado en ';'.

Una cabecera de un subprograma comienza por la palabra reservada `SUBPROGRAMA` seguida del identificador del subprograma, de la declaración de parámetros formales, de la declaración opcional del tipo del valor devuelto por el subprograma y finaliza en ';'. El tipo del valor devuelto se declara especificando la secuencia: del símbolo ':' seguido de un identificador de tipo.

La declaración de parámetros formales es opcional, pero si aparece la declaración de parámetros va delimitada entre los paréntesis de abrir y cerrar. Una declaración de parámetros consiste en la definición de una lista de identificadores no vacía separados por el carácter ',' seguida del separador ':' y finalizada por un identificador de tipo.

Un identificador de tipo sólo puede ser alguna de las siguientes palabras reservadas: `integer, boolean, double, char o String.`

El cuerpo del programa y el cuerpo de un subprograma tienen la misma estructura, finalizando el primero en un '.'y el segundo en un ';' como ya se ha comentado anteriormente.

Un cuerpo se define delimitado entre las palabras reservadas `INICIO` y `FIN` que se darán siempre aunque el cuerpo de sentencias ejecutables está vacío. Un cuerpo es una lista secuencial de diferentes clases de sentencias que se definen a continuación que pueden aparecer en cualquier orden y cantidad con la excepción que se comenta más adelante relativa a la sentencia `return`:

Las clases de sentencias que pueden formar parte de un cuerpo pueden ser de : asignación, llamada a subprograma, lectura , escritura y de devolución de control. Toda sentencia finaliza en un ';'.

- ✓ *Asignación* – se forma con un identificador de variable seguido del operador de asignación, representado por la secuencia de símbolos ´:=´, seguido por una expresión.
- ✓ *Devolución de control* - sólo puede darse una sentencia de devolución en el cuerpo de un subprograma o programa y de existir tendrá que ser la última sentencia del cuerpo. Se forma con la palabra reservada `return` seguida de una expresión.
- ✓ *Sentencia de llamada a subprograma* - se forma con un identificador seguido de 0 o más expresiones separadas por el símbolo coma ',' delimitadas por los paréntesis ´(´ y ´)´, símbolos que son obligatorios si existen argumentos en la llamada.
- ✓ *Sentencia de lectura* - se forma con la palabra reservada `LEER` seguida de una lista de identificadores de variables no vacía delimitada entre paréntesis.
- ✓ *Sentencia de escritura* – se forma con la palabra reservada `ESCRIBIR` seguida de una lista de expresiones de cualquier tipo, que puede estar vacía, delimitada entre paréntesis que siempre deben aparecer.

Las **expresiones** deben cumplir con las siguientes características:

□ Los **operandos simples** pueden ser constantes enteras, reales, carácter, lógicas, literal cadena de caracteres, o identificadores o bien otra expresión delimitada por paréntesis.

- ✓ Las constantes enteras, constan de al menos un dígito del 0 al 9.
- ✓ Las constantes reales, se construyen escribiendo una secuencia de dígitos no vacía separada por el símbolo '.' de otra secuencia de dígitos no vacía.
- ✓ Literal carácter, es cualquier carácter excepto el carácter apóstrofo delimitado por apóstrofos, sea como ejemplo 'a', o bien el carácter apóstrofo representado por la secuencia \' delimitado entre apóstrofos.
- ✓ Literal cadena de caracteres, cero o más caracteres delimitados por el carácter comillas dobles teniendo en cuenta que dentro de la cadena el propio carácter de comillas dobles se debe representar por la secuencia \" y el carácter tabulador por la secuencia \t. Un literal cadena de caracteres no puede ocupar más de una línea.
- ✓ Literal lógico, es alguno de los valores lógicos representados por las palabras reservadas `TRUE` o `FALSE`.
- ✓ Un identificador se forma comenzando por un símbolo alfabético, seguido de un número indeterminado, cero o más, de símbolos alfanuméricos o símbolo de subrayado.

□ Los **operadores** que pueden aparecer en una expresión son:

- aritméticos binarios: ´ +´ (suma), ´-´ (resta), ´*´ (producto), ´/´ (cociente).
- de relación: == (igual), <> (distinto), < (menor), > (mayor),<= (menoroigual), >= (mayoroigual).
- lógicos binarios: OR (o lógico) y AND (y lógico)

Todos los operadores binarios aritméticos y lógicos tienen asociatividad a derechas, los de relación no son asociativos.

La precedencia de los operadores anteriores es la siguiente:

1º- AND, *, /

2º- OR, +, -

3º- <, <=, >, >=, ==, <>

En L se pueden definir comentarios en línea - comienzan por el par de caracteres / / seguidos de cualquier secuencia de caracteres que no puede incluir el fin de línea.

El siguiente ejemplo muestra una especificación en L sintácticamente válida pero incorrecta semánticamente,

```
PROGRAMA ejemplo ;
CONSTANTES
    entera = 01;
    real = 0.0;
    cierto = TRUE;
    blanco = " ";
    comillas = "\"";
     no = '\';
VARIABLES
    x:=0,y,z:=" ": String;
    x:=0,y,z:=' ':char;
SUBPROGRAMA primero : char ;
INICIO
return TRUE;
FIN;
SUBPROGRAMA segundo (par_1,par_2:String) ;
INICIO
LEER(uno_1,dos2);
FIN ;

INICIO
LEER(uno_1,dos2);
LEER ( uno //- es un comentario hasta final de línea
);
Uno := dos*10.0*TRUE*tres+0.0+FALSE;

ESCRIBIR();
ESCRIBIR (dos* (10.0+0+0.0+
          tres),
         uno );

/////////-
LEER ( uno);
return FALSE;
FIN.
```

Desarrollo del analizador. Nombres de los ficheros

Se trata de escribir una especificación en `JavaCC` para obtener un analizador léxico-sintáctico del lenguaje L.

Para que la corrección de la práctica pueda llevarse a cabo es **imprescindible** que se usen los siguientes nombres:

`Gramatica.jj`, nombre del fichero que contiene la especificación escrita en JavaCC

`Analizador`, nombre para la especificación escrita en JavaCC

`prueba1`, `prueba2`, … nombres, **sin extensión**, para los diferentes ficheros de pruebas

En el caso de que no se detecten errores léxicos o sintácticos en la salida aparecerá la siguiente información:

Fichero analizado

No se han detectado errores léxico-sintácticos

Si el programa tiene errores, se hará un tratamiento simple que es el tratamiento por defecto correspondiente al funcionamiento del analizador generado por JavaCC.

El nombre de los ficheros de entrada y salida para la ejecución del analizador generado deben proporcionarse como argumentos en la línea de comandos.

Documentación

Es obligatorio presentar la práctica acompañada de una documentación por escrito que incorporará un índice con la relación de los apartados que la componen. No se admiten documentaciones escritas a mano. Entre los puntos incluidos deben estar:

- una portada indicando el nombre del alumno, número de matrícula, grupo de clase y profesor correspondiente al grupo,
- una memoria con ***extensión máxima de una página*** justificando aquellos aspectos de la solución que se consideren más significativos,
- la especificación escrita en JavaCC; este punto deberá consistir en una ***transcripción literal del código*** (en texto plano y sin editar); será imprescindible para aprobar la práctica que haya una correspondencia exacta entre los ficheros de código y su versión incorporada en la documentación,
- un apartado con ***extensión máxima de dos páginas*** dedicado a las pruebas del programa obtenido en el que se comente cómo se han organizado las pruebas y qué características se han pretendido verificar en cada uno de los ficheros de pruebas usados. El contenido de los ficheros de prueba no se incluirá en la documentación.

En este apartado NO pueden incluirse como pruebas el ejemplo citado en este enunciado.

Evaluación

Esta actividad práctica representa el 10% de la nota final del proceso de evaluación continua y se debe realizar obligatoriamente y aprobar (5,0) para que la calificación global de la asignatura resulte superada.

Para la evaluación será imprescindible presentar todos los documentos que se solicitan en el apartado de documentación.

En el proceso de evaluación el analizador obtenido debe funcionar correctamente. Para la evaluación es imprescindible presentar todos los apartados que se piden en la documentación y se tendrá en cuenta la calidad y la estructuración de la documentación. Dentro del proceso de evaluación de esta actividad se podrá llamar al alumno para que defienda presencialmente su solución. Si se detecta una copia la práctica estará suspensa.

En la evaluación se valorará colateralmente también la competencia transversal Razonamiento Crítico.

Presentación y plazo

El ejercicio resuelto se **entregará al profesor del grupo al que se pertenece**, de la siguiente manera:

- la especificación JavaCC y los ficheros de prueba relacionados en la documentación, por correo electrónico **antes de** entregar la documentación por escrito,
- la documentación por escrito, **personalmente en el Despacho, en horas de tutorías.**

El **plazo** de presentación para la convocatoria de junio termina el día **18 de mayo de 2018.**