Procesadores del Lenguaje

Curso 2020-2021

Práctica – Segunda Aproximación.

Traductor de Expresiones a Notación Prefija

Esta segunda parte de la práctica consiste en transformar el traductor de la sesión previa. El traductor previo seguía un esquema conocido como traducción directa, es decir, la conversión e impresión de la entrada analizada se hace sobre la marcha al mismo tiempo que se explora el árbol de derivación.

En la nueva propuesta aplicaremos un esquema de traducción algo más elaborado, aunque menos sofisticado que el uso de Árboles Sintácticos Abstractos, que se verán más adelante.

Para esta ocasión emplearemos también una entrada de expresiones en notación infija, pero la salida será en notación prefija con paréntesis.

Trabajo a realizar:

Para ésta aproximación se pide únicamente la traducción de expresiones aritméticas y de asignaciones a variables de una notación infija como la que usamos habitualmente en el Lenguaje C a notación prefija.

1. Leed el enunciado completo antes de comenzar.
2. Utilizad el fichero proporcionado calc8.y
3. Adaptad la semántica (y la gramática sólo cuando sea necesario) para que traduzca las expresiones introducidas a notación prefija. Seguid la secuencia de desarrollo propuesta en las especificaciones (al final del enunciado).

Entrega:
Subid el fichero `calc8.y`.

Incluid en la cabecera del fichero dos líneas de comentario iniciales. La primera con vuestros nombres y número de grupo. Y la segunda con los correos electrónicos (separados con un espacio).

Entregad también un documento llamado calc8.pdf con una breve explicación del trabajo realizado.

Después de hacer la entrega, descargad vuestro fichero y comprobad que funciona correctamente y que cumple con las indicaciones.

Sobre la notación prefija

Una expresión en notación infija como `1+2*3` se representa en notación prefija como: `(+ 1 (* 2 3))`. La conversión debe tener en cuenta la precedencia y asociatividad que definen las convenciones.

En la notación prefija, los paréntesis sirven para establecer la relación de los operadores con sus respectivos operandos. Al mismo tiempo, determinan con qué precedencia y asociatividad se evalúan los operadores.

Una expresión como `(* (+ 1 2) 3)` se entiende que hace referencia a la equivalente en notación infija: `(1+2)*3` en la que se ha forzado una precedencia alternativa a la convencional.

Una sentencia de asignación entre variables y expresiones como `a = b*7 + c ;` debe traducirse como `(= a (+ (* b 7) c))`

Especificaciones

Proponemos una secuencia de pasos para abordar esta práctica:

1. Estudiad detenidamente el ejercicio resuelto de conversión entre notaciones con *código diferido*. Comprobad las estructuras de datos y las funciones que se usan.
2. Estudiad el código **`calc8.y`** proporcionado:
    a. la estructura de datos para los *Token* (**`%union`**).
    b. la directiva **`%type`** asocia un campo determinado de la **`union`** a cada uno de los *No Terminales* y *Token* que intervienen en la gramática y que deben transmitir un valor hacia arriba en el árbol sintáctico.
    c. qué funciones coinciden con las incluidas en el ejercicio resuelto.
    d. cómo opera la función **`yylex()`**.
3. No modifiquéis **`yylex()`**.
4. Incluid las acciones semánticas adecuadas en las producciones para que la salida pase de infija a prefija. En este caso, no se debe imprimir nada en la salida estándar en los nodos inferiores. La salida debe ser añadida en el campo **`$$.cadena`** para transmitirla hacia arriba. La impresión de la traducción debe hacerse en el axioma. Dado que **`%type`** asocia un campo determinado de la **`union`** a los *No Terminales*, se podrá operar con **`$$`** en vez de con **`$$.cadena.`** La entrada **`1+2*3`** debería traducirse como **`(+ 1 (* 2 3))`**. Prestad atención a la precedencia y asociatividad definidas en la cabecera del fichero calc8.y .
5. Abordad la traducción de las *Variables*, tanto en la parte de componentes de una expresión, como receptoras de una asignación.
6. Prestad atención al diseño jerárquico de la gramática. Debería haber una *Sentencia* que deriva en *Expresion* por un lado y en *Asignacion* por otro. La traducción debería volcarse a la salida a nivel de *Sentencia*. Para separar sentencias usaremos saltos de línea.