Universidad de La Laguna

E. T. S. DE INGENIERÍA INFORMÁTICA
Departamento de E.I.O. y Computación
Compiladores

Traducción dirigida por la sintaxis

Factor de ponderación [0-10]: 10

6.1. Introducción

La siguiente gramática:

$Expr \rightarrow Expr$ **+** $Term|Expr$ **-** $Term|Term$
$Term \rightarrow Term$ ***** $Factor|Term$ **/** $Factor|Factor$
$Factor \rightarrow$**(**$Expr$**)** | **-**$Factor|$ **id** | **num**

genera expresiones aritméticas con operadores suma, resta, producto, división y menos unario. El conjunto de símbolos no terminales es $V = \{Expr, Term, Factor\}$ y los tokens son: $\Sigma = \{+, -, *, /,), (, id, num\}$. Supondremos que los identificadores (**id**) son los identificadores válidos en *Pascal-* y que **num** son números enteros.

La definición dirigida por sintaxis de la tabla 6.1 está basada en la gramática anterior y utiliza el atributo *ptr* (puntero a un nodo) y las funciones *mknode*, *mkleaf* y *mkunode* para crear dinámicamente nodos del árbol de análisis correspondiente a la expresión aritmética.

La función *mknode(a, b, c)* crea en memoria dinámica un nodo que tiene 'a' como operando y 'b' y 'c' como punteros a los nodos que contienen los operadores de este operando.

Las funciones *mkleaf* y *mkunode* crean respectivamente nodos hojas (correspondientes a un identificador o a un número) y nodos correspondientes al operador unario y operan de forma equivalente.

6.2. Descripción

La práctica a realizar consiste en:

1. Modificar el analizador léxico que ha utilizado en prácticas anteriores para que reconozca los elementos léxicos de esta gramática.

2. Partiendo de la definición dirigida por la sintaxis de la tabla 6.1, producir un esquema de traducción.

$Expr \rightarrow Expr_1 + Term$	$Expr.ptr := mknode('+', Expr_1.ptr, Term.ptr)$
$Expr \rightarrow Expr_1 - Term$	$Expr.ptr := mknode('-', Expr_1.ptr, Term.ptr)$
$Expr \rightarrow Term$	$Expr.ptr := Term.ptr$
$Term \rightarrow Term_1 * Factor$	$Term.ptr := mknode('*', Term_1.ptr, Factor.ptr)$
$Term \rightarrow Term_1/Factor$	$Term.ptr := mknode('/', Term_1.ptr, Factor.ptr)$
$Term \rightarrow Factor$	$Term.ptr := Factor.ptr$
$Factor \rightarrow (Expr)$	$Factor.ptr := Expr.ptr$
$Factor \rightarrow -Factor$	$Factor.ptr := mkunode('-', Factor.ptr)$
$Factor \rightarrow id$	$Factor.ptr := mkleaf(id, id.ptr)$
$Factor \rightarrow num$	$Factor.ptr := mkleaf(id, id.ptr)$

Cuadro 6.1: Definición dirigida por la sintaxis

3. Eliminar la recursividad por la izquierda en el esquema de traducción resultante.

4. Construir un analizador sintáctico descendente recursivo predictivo, con recuperación de errores que al evaluar las acciones semánticas asociadas con las producciones de la gramática produzca como resultado la construcción del árbol sintáctico de la expresión.

5. Diseñar funciones que implementen el recorrido del árbol sintáctico de la expresión en inorden, preorden y postorden.