Práctica 5 - Recursividad

Cesar Padilla Lazos

Wing Manuel Ortiz Uribe

Objetivo: Revisar la gramática de su propuesta de lenguaje para asegurar la generación de árboles sintácticos por el método descendente.

Instrucciones:

1. Escriba la gramática completa de su lenguaje y valide que pueda generar todas las cadenas que desea analizar en su código.

DECLARACION -> **VARIABLE** TIPO ID **=** VALOR;

TIPO -> **ENTERO** | **DECIMAL** | **TEXTO** | **BOOLEANO**

VALOR -> ENTERO | DECIMAL | TEXTO | VERDADERO | FALSO

IF-> **SI?(**EVALUACION**){**INSTRUCCION**}** ELSE

EVALUACION -> OPERANDO OPERADOR_COMP OPERANDO;

OPERANDO -> ID | VALOR

OPERADOR_COMP -> **!=** | **==** | **<=** | **>=** | **<** | **>**

INSTRUCCION -> COD; INSTRUCCION | EPSILON

COD -> IMPRIMIR | ASIGNACION

ASIGNACION -> ID = OPC; | ID = OPC O;

OPC -> ID | VALOR

O -> + ID | - ID | + VALOR | - VALOR | O

ELSE -> **DE_LO_CONTRARIO{**INSTRUCCION**} |** EPSILON

_IMPRIMIR -> **IMPRIMIR(**OPC**)**;

2. Factorice la gramática y especifique cuáles sentencias se “redujeron” o aclare que no fue necesario demostrando por qué.

ASIGNACION -> ID = OPC; | ID = OPC O;

ASIGNACION -> ID = OPC ASIGNACION’;

ASIGNACION’ -> O | EPSILON

OPC -> ID | VALOR

O -> + OPC | - OPC | O

O -> O’ OPC | O

O’ -> + | -

3. Elimine la recursividad por la izquierda y escriba la gramática ampliada. Si no presenta recursividad especifíquelo.

Sin recursividad en la gramática

4. Asegúrese de que la gramática final no es ambigua eligiendo dos sentencias: una sencilla y una semi compleja (condición, repetición) para su representación. Aunque eso no lo asegure del todo, será suficiente para la práctica.

VARIABLE ENTERO num2 = 2;

VARIABLE TIPO ID = VALOR

```
SI? (num1 == num2){
   IMPRIME('HOLA')
}
```

SI?(OPERANDO OPERADOR_COMP OPERANDO**){**
    _IMPRIMIR(OPC)
}

5. Entregue en este documento los elementos de cada uno de los puntos, el punto inicial requiere demostración, solo especificar su gramática.