**EJERCICIO**

Diseñar el comprobador de tipos de lenguaje Simple como un **Esquema de Traducción** que considere las siguientes reglas de compatibilidad de tipos:

1. No se permite la redeclaración de identificadores
2. En la asignación se permite lo siguiente:

|  |  |
| --- | --- |
| Id | expresion |
| entero | entero |
| real | real |
| Real | entero |
| Carácter | carácter |

Considera la siguiente modificación a la gramática del lenguaje que agrega la producción E 🡪 literal

P 🡪 V C

V 🡪 **id :** T V | **empty**

T 🡪 **entero** | **real** | **caracter**

C 🡪 **inicio** S **fin**

S 🡪 **id opasig** E S | **empty**

E 🡪 **id** | **num** | **num.num | literal**

Las constantes literal se consideran tipo “caracter”.

**DESARROLLO:**

**Esquema de Traducción**

P 🡪 V C **{1}**

V 🡪 **id :** T **{6}** V1 **{7}**| **empty {2}**

T 🡪 **entero {3}** | **real {4}** | **carácter {5}**

C 🡪 **inicio** S **fin {8}**

S 🡪 **id opasig** E **{14}** S1 **{15}**| **empty {9}**

E 🡪 **id {10}**| **num {11}**| **num.num {12}| literal {13}**

**Tabla de acciones semánticas**

|  |  |
| --- | --- |
| **No** | **ACCION SEMANTICA** |
| 1 | P.tipo := if V.tipo == VACIO AND C.tipo == VACIO then  VACIO  else  ERROR\_tipo |
| 2 | V.tipo := VACIO |
| 3 | T.tipo := ‘entero’ |
| 4 | T.tipo := ‘real’ |
| 5 | T.tipo := ‘caracter’ |
| 6 | if buscaTipo ( id.entrada ) == nil then  añadeTipo ( id.entrada, T.tipo )  V.tipoaux := VACIO  else  ERROR\_TIPO |
| 7 | V.tipo := if V.tipoaux == VACIO AND V1.tipo == VACIO then  VACIO  else  ERROR\_TIPO |
| 8 | C.tipo := S.tipo |
| 9 | S.tipo := VACIO |
| 10 | E.tipo := buscatipo(id.entrada) |
| 11 | E.tipo := ‘entero’ |
| 12 | E.tipo := ‘real’ |
| 13 | E.tipo := ‘caracter’ |
| 14 | if buscaTipo ( id.entrada ) == E.tipo OR ( buscaTipo(id.entrada) == ‘real’ AND E.tipo == ‘entero’) then  S.tipoaux := VACIO  else  ERROR\_TIPO |
| 15 | S.tipo := if S.tipoaux == VACIO and S1.tipo == VACIO then  VACIO  else  ERROR\_TIPO |

Con el comprobador de tipos de la p.25 analizar el siguiente programa fuente de entrada:

m : integer ;

n : integer ;

n := ( 6 mod m ) + 2 \* ( m + 1 )

1. **ANALISIS LEXICO:**

id1 : integer ;

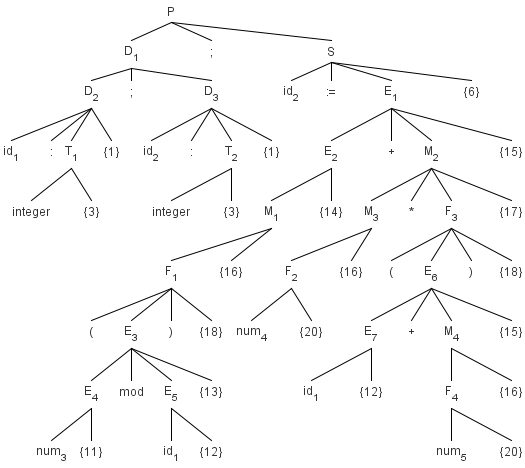
id2 : integer ;

id2 := ( num3 mod id1 ) + num4 \* ( id1 + num5 ) ;

**TABLA DE SIMBOLOS**

|  |  |  |  |
| --- | --- | --- | --- |
| **Entrada** | **Lexema** | **Complex** | **Tipo** |
| 1 | m | id |  |
| 2 | n | id |  |
| 3 | 6 | num |  |
| 4 | 2 | num |  |
| 5 | 1 | num |  |

1. **ANALISIS SINTACTICO - SEMANTICO:**

****

**RECORRIDO DEL ARBOL (COMPROBADOR DE TIPOS)**

**TABLA DE SIMBOLOS**

|  |  |  |  |
| --- | --- | --- | --- |
| **Entrada** | **Lexema** | **Complex** | **Tipo** |
| 1 | m | id | integer |
| 2 | n | id | integer |
| 3 | 6 | num |  |
| 4 | 2 | num |  |
| 5 | 1 | num |  |

|  |  |
| --- | --- |
| **ATRIBUTOS** | |
| **Simbolo** | **Tipo** |
| P |  |
| D1 |  |
| D2 |  |
| T1 | integer |
| D3 |  |
| T2 | integer |
| S | **VACIO** |
| E1 | integer |
| E2 | integer |
| M1 | integer |
| F1 | integer |
| E3 | integer |
| E4 | integer |
| E5 | integer |
| M2 | Integer |
| M3 | integer |
| F2 | integer |
| F3 | integer |
| E6 | integer |
| E7 | integer |
| M4 | integer |
| F4 | integer |

R = Por lo tanto, **SI hay compatibilidad de tipos** en el programa fuente ingresado