Semántica

<programa>::= <sentencia>”;”<programa2>

evalprograma(árbol, ts, errorStatus)

if (not(errorStatus)) then

evalSentencia(arbol^.hijos[1], ts, errorStatus);

evalprograma2(arbol^.hijos[3], ts, errorStatus);

<programa2>::= <programa>|ε

evalprograma2(árbol, ts, errorStatus)

if arbol^.hijos[1]^.simbolos=programa then

evalprograma(arbol^.hijos[1], ts, errorStatus);

<sentencia>::= <asig>|<lectura>|<escritura>|<condicional>|<ciclo>

evalSentencia(arbol, ts, errorStatus)

if (not(errorStatus)) then

if arbol^.hijos[1]^.simbolos=asig then

evalasig(arbol^.hijos[1], ts, errorStatus)

else if arbol^.hijos[1]^.simbolos=lectura then

evalLectura(arbol^.hijos[1], ts, errorStatus)

else if arbol^.hijos[1]^.simbolos=escritura then

evalEscritura(arbol^.hijos[1], ts, errorStatus)

else if arbol^.hijos[1]^.simbolos=condicional then

evalcondicional(arbol^.hijos[1], ts, errorStatus)

else if arbol^.hijos[1]^.simbolos=ciclo then

evalciclo(arbol^.hijos[1], ts, errorStatus)

<asig>::= ”id” ”opasig”<expresion>

evalAsig(árbol, ts, errorStatus)

if (not(errorStatus)) then

newTResultado(resultado);

evalExpresion(arbol^.hijos[3], ts, resultado, errorStatus);

asignar(ts, arbol^.hijos[1]^.lexema, resultado, errorStatus);

<expresion>::= <explista>|<exparit>

evalExpresion(árbol, ts, resultado, errorStatus)

if (not(errorStatus)) then

if arbol^.hijos[1]^.simbolos=exparit then

evalexparit(arbol^.hijos[1], ts, resultado, errorStatus)

else if arbol^.hijos[1]^.simbolos=explista then

evalexplista(arbol^.hijos[1], ts, resultado, errorStatus); //esto lista

<explista>::= <oplista>|<lista>

evalexplista(árbol, ts, resultado, errorStatus)

if (not(errorStatus)) then

if arbol^.hijos[1]^.simbolos=oplista then

evaloplista(arbol^.hijos[1], ts, resultado, errorStatus)

else if arbol^.hijos[1]^.simbolos=lista then

evallista(arbol^.hijos[1], ts, resultado, errorStatus);

<oplista>::= ”rest” ”(“ <explistaoid> ”)” |”cons” “(“ <exparit> ”,” <explistaoid> ”)”

evaloplista(árbol, ts, resultado, errorStatus)

if (not(errorStatus)) then

if arbol^.hijos[1]^.simbolos=rest then

newTResultado(resultadoLista);

evalexplistaoid(arbol^.hijos[3], ts, resultadoLista, errorStatus);

resultado.lista:= RestL(resultadoLista.lista);

else if arbol^.hijos[1]^.simbolos=cons then

newTResultado(resultadoLista);

newTResultado(resultadoArit);

evalexplistaoid(arbol^.hijos[5], ts, resultadoLista, errorStatus);

evalexparit(arbol^.hijos[3], ts, resultadoArit, errorStatus);

if resultadoArit.numero=-1 then

errorStatus:=true;

ConsL(resultadoLista.lista, resultadoArit.numero);

resultado.lista:= resultadoLista.lista;

<explistaoid>::= <explista> | “id”

evalexplistaoid(árbol, ts, resultado, errorStatus)

if (not(errorStatus)) then

if arbol^.hijos[1]^.simbolos=explista then

evalexplista(arbol^.hijos[1], ts, resultado, errorStatus);

else

resultado:= obtenervalor(ts, arbol^.hijos[1]^.lexema, errorStatus);

<lista>::= ”[“<listanum>”]”

evallista(árbol, ts, resultado, errorStatus)

if (not(errorStatus)) then

evallistanum(arbol^.hijos[2], ts, resultado, errorStatus);

<listanum>::= ”consent”<listanum2>

evallistanum(árbol, ts, resultado, errorStatus)

if (not(errorStatus)) then

val(arbol^.hijos[1]^.lexema, numero, codigoerror);

if codigoerror = 0 then

punteroAux:= resultado.lista.cab; // asignacion al final de la lista

punteroAnt:= nil;

while punteroAux <> nil do

punteroAnt:=punteroAux;

punteroAux:=punteroAux^.sig;

New(punteroNodoAux);

punteroNodoAux^.info:=numero;

punteroNodoAux^.sig:=nil;

if punteroAnt=nil then

resultado.lista.cab:=punteroNodoAux;

inc(resultado.lista.tam);

else

punteroAnt^.sig:=punteroNodoAux;

inc(resultado.lista.tam);

resultado.isReal:=false;

evallistanum2(arbol^.hijos[2], ts, resultado, errorStatus);

<listanum2>::= ”,”<listanum>| ε

evallistanum2(árbol, ts, resultado, errorStatus)

if (not(errorStatus)) then

if arbol^.hijos[1]^.simbolos=coma then

evallistanum(arbol^.hijos[2], ts, resultado, errorStatus);

<exparit>::= ”first” “(“ <explistaoid> “)”<exparit2> | ”consent”<exparit2> | ”id”<exparit2> | ”(“<exparit>”)”<exparit2>

evalExparit(árbol, ts, resultado, errorStatus)

if (not(errorStatus)) then

if debugMode then writeln('El simbolo es: ' + stringSimbolos[arbol^.hijos[1]^.simbolos]);

if arbol^.hijos[1]^.simbolos=consent then

val(arbol^.hijos[1]^.lexema, numero, codigoerror)

if codigoerror = 0 then

newTResultado(res);

resultado.numero:=numero;

resultado.isReal:=true;

evalexparit2(arbol^.hijos[2], ts, res,resultado, errorStatus);

else if arbol^.hijos[1]^.simbolos=id then

newTResultado(res);

Resultado:= obtenervalor(ts, arbol^.hijos[1]^.lexema, errorStatus);

if not(errorStatus) then evalexparit2(arbol^.hijos[2], ts, res,resultado, errorStatus);

else if arbol^.hijos[1]^.simbolos=parentesis1 then

newTResultado(res);

evalexparit(arbol^.hijos[2], ts,resultado, errorStatus);

evalexparit2(arbol^.hijos[4], ts, res,resultado, errorStatus);

else if arbol^.hijos[1]^.simbolos=first then

newTResultado(res);

newTResultado(resultadoLista);

evalexplistaoid(arbol^.hijos[3], ts, resultadoLista, errorStatus);

FirstL(resultadoLista.lista, x);

resultado.numero:=x;

resultado.isReal:=true;

evalexparit2(arbol^.hijos[5], ts, res,resultado, errorStatus);

<exparit2>::= “oparit”<exparit>| ε

evalExparit2(árbol, ts, res, resultado, errorStatus)

if (not(errorStatus)) then

if arbol^.hijos[1]^.simbolos=oparit then

evalexparit(arbol^.hijos[2], ts, res, errorStatus);

if arbol^.hijos[1]^.lexema = '+' then resultado.numero:=resultado.numero + res.numero

else if arbol^.hijos[1]^.lexema = '-' then resultado.numero:=resultado.numero - res.numero

else if arbol^.hijos[1]^.lexema = '\*' then resultado.numero:=resultado.numero \* res.numero

else if arbol^.hijos[1]^.lexema = '/' then resultado.numero:=resultado.numero div res.numero;

<lectura>::= <leerE>|<leerL>

evalLectura(arbol, ts, errorStatus)

if (not(errorStatus)) then

if arbol^.hijos[1]^.simbolos=leerE then

evalLeerE(arbol^.hijos[1], ts, errorStatus)

else if arbol^.hijos[1]^.simbolos=leerL then

evalLeerL(arbol^.hijos[1], ts, errorStatus);

<leerE>::= “leerEntero” “(“ “cadena” “,” “id” “)”

evalLeerE(árbol, ts, errorStatus)

if (not(errorStatus)) then

write(copy(arbol^.hijos[3]^.lexema, 2, Length(arbol^.hijos[3]^.lexema)-2));

readln(X);

for i := 1 to Length(X) do

case X[i] of

'0'..'9': ;

else

errorStatus:=true;

if (not(errorStatus)) then

resultado.numero:=StrToInt(X);

resultado.isReal:=true;

asignar(ts, arbol^.hijos[5]^.lexema, resultado, errorStatus);

<leerL>::= “leerLista” “(“ “cadena” “,” “id” “)”

evalLeerL(árbol, ts, errorStatus)

if (not(errorStatus)) then

write(copy(arbol^.hijos[3]^.lexema, 2, Length(arbol^.hijos[3]^.lexema)-2));

readln(X);

if ((X[1] = '[') and (X[Length(X)] = ']') and (X[2] <> ',') and (X[Length(x) -1] <> ',')) then

if (X <> '[]') then

for i := 2 to (Length(X) -1) do

case X[i] of

'0'..'9': begin end;

',': begin end;

else

errorStatus:=true;

else

errorStatus:=true;

if (not(errorStatus)) then

resultado.isReal:=false;

resultado.lista:=ParseLista(X, errorStatus);

asignar(ts, arbol^.hijos[5]^.lexema, resultado, errorStatus);

<escritura>::= “escribir” “(“ “cadena” “,” <expresión> “)”

evalEscritura(árbol, ts, errorStatus)

if (not(errorStatus)) then

newTResultado(resultado);

evalExpresion(arbol^.hijos[5], ts, resultado, errorStatus);

if not(errorStatus) then

write(copy(arbol^.hijos[3]^.lexema, 2, Length(arbol^.hijos[3]^.lexema)-2));

writeln(TResultadoToString(resultado));

<condicional>::= “si”<condicion> “entonces”<programa><condicional1>

evalcondicional(árbol, ts, errorStatus)

if (not(errorStatus)) then

evalcondicion(arbol^.hijos[2], ts, estado, errorStatus);

if estado=true then

evalprograma(arbol^.hijos[4], ts, errorStatus)

else

evalcondicional1(arbol^.hijos[5], ts, errorStatus);

<condicional1>::= “fin” |”sino”<programa>”fin”

evalcondicional1(arbol, ts, errorStatus)

if (not(errorStatus)) then

if arbol^.hijos[1]^.simbolos=sino then

evalprograma(arbol^.hijos[2], ts, errorStatus);

<condicion>::= <exparit> “oprel” <exparit> | ”null” “(“ <explistaoid> “)”

evalcondicion(árbol, ts, estado, errorStatus)

if (not(errorStatus)) then

if arbol^.hijos[1]^.simbolos = exparit then

newTResultado(resultado);

newTResultado(res);

evalexparit(arbol^.hijos[1], ts, resultado, errorStatus);

evalexparit(arbol^.hijos[3], ts, res, errorStatus);

if arbol^.hijos[2]^.lexema = '=' then estado:=resultado.numero=res.numero

else if arbol^.hijos[2]^.lexema = '<' then estado:=resultado.numero<res.numero

else if arbol^.hijos[2]^.lexema = '<=' then estado:=resultado.numero<=res.numero

else if arbol^.hijos[2]^.lexema = '>' then estado:=resultado.numero>res.numero

else if arbol^.hijos[2]^.lexema = '>=' then estado:=resultado.numero>=res.numero

else if arbol^.hijos[2]^.lexema = '<>' then estado:=resultado.numero<>res.numero

else if arbol^.hijos[1]^.simbolos=null then

evalexplistaoid(arbol^.hijos[3], ts, resultado, errorStatus);

if ListaVacia(resultado.lista) then estado:=true else estado:=false;

<ciclo>::= “mientras”<condicion>”hacer”<programa>”fin”

evalciclo(árbol, ts, errorStatus)

if (not(errorStatus)) then

evalcondicion(arbol^.hijos[2], ts, estado, errorStatus);

while estado = true do

evalprograma(arbol^.hijos[4], ts, errorStatus);

evalcondicion(arbol^.hijos[2], ts, estado, errorStatus);