# **Acciones Semánticas**

**Pseudocódigo**

S -> prog {

printf("Compilacion OK\n");

guardarArbol(ptrPrograma);

generarAssembler(&ptrPrograma);

};

Prog -> sent {

printf("Regla 1 (PROG->SENT)\n");

ptrPrograma = ptrSentencia;

}

| prog sent {

printf("Regla 2 (PROG -> PROG SENT)\n");

ptrProg = crearNodo(“subArbol\_n°”, ptrProg, ptrSent);

};

Sent -> read {

printf("Regla 3 (SENT -> READ)\n");

ptrSent = crearNodo("union",ptrRead, ptrIf);

}

| write {

printf("Regla 3 (SENT -> WRITE)\n");

ptrSent = ptrWrite;

}

| asig {

printf("Regla 3 (SENT -> ASIG)\n");

ptrSent = ptrAsig;

};

Read -> READ ID {

printf("Regla 4 (READ -> read id)\n");

ptrCond = crearNodo("<", crearHoja($2), crearHoja("\_1"));

ptrIf = crearNodo("IF", ptrCond, crearHoja("@msgMayor\_1"));

ptrRead = crearNodo("READ",crearHoja($2), NULL);

};

Asig -> ID { strcpy(varString, $1);} ASIGNA posicion {

printf("Regla 5 (ASIG -> id asigna POSICION)\n");

if(listaVacia == false){

ptrUnion = crearNodo("=",crearHoja(varString), crearHoja("@aux"));

ptrAsig = crearNodo("unionAsig", ptrPosicion, ptrUnion);

} else{

ptrAsig = crearNodo("=", crearHoja(varString),ptrPosicion); }

};

Posición -> POSICION PARA ID PYC CA lista CC PARC {

printf("Regla 6 (POSICION -> posicion para id pyc ca LISTA cc parc)\n");

ptrFalse = crearHoja("@msgNoEncontro");

ptrCuerpoAnt = vectorCuerpo[cont-1];

(ptrCuerpoAnt)->ptrDer = ptrFalse;

ptrPosicion = ptrLista;

}

| POSICION PARA ID PYC CA CC PARC {

printf("Regla 7 (POSICION -> posicion para id pyc ca cc parc)\n");

listaVacia = true;

ptrPosicion = crearHoja("@msgListaVacia");

};

//Pos empieza en 0

//Cont empieza en 0

Lista -> CTE {

printf("Regla 8 (LISTA -> CTE)\n");

pos++;

ptrCond = crearNodo("==", crearHoja("pivot"),crearHoja($1));

ptrTrue = crearNodo("=", crearHoja("@aux"),crearHoja(pos));

ptrCuerpo = crearNodo("CUERPOIF", ptrTrue, NULL);

vectorCuerpo[cont] = ptrCuerpo;

cont++;

ptrIf = crearNodo("IF", ptrCond, ptrCuerpo);

ptrLista = ptrIf;

}

| lista COMA CTE {

printf("Regla 9 (LISTA -> LISTA coma cte)\n");

pos++;

ptrCond = crearNodo("==", crearHoja("pivot"),crearHoja($3));

ptrTrue = crearNodo("=", crearHoja("@aux"),crearHoja(pos));

ptrCuerpo = crearNodo("CUERPOIF", ptrTrue, NULL);

ptrIf = crearNodo("IF", ptrCond, ptrCuerpo);

ptrCuerpoAnt = vectorCuerpo[cont-1];

(ptrCuerpoAnt)->ptrDer = ptrIf;

vectorCuerpo[cont] = ptrCuerpo;

cont++;

};

Write -> WRITE CTE\_S {

printf("Regla 10 (WRITE -> write cte\_s)\n");

ptrWrite = crearNodo("WRITE",crearHoja($2),NULL);

}

| WRITE ID {

printf("Regla 11 (WRITE -> write id)\n");

ptrWrite = crearNodo("WRITE",crearHoja($2),NULL);

};