SZIE eta murriztapen semantikoak inplementatzeko laborategia:

1- zerorekin zatiketa

```
main_prog → func main () { stmts }


stmts   →   stmt ;
            | stmts stmt ;

stmt    →   id = expr
            | if { stmts }

expr →        expr + expr
            | expr - expr
            | expr * expr
            | expr / expr
            | expr < expr
            | expr <= expr
            | expr > expr
            | expr >= expr
            | id
            | num_integer
            | num_float
```

ATRIBUTUAK:

Lexikoak:

  id.izena, num_integer.izena, num_float.izena: dagokien karaktere katea

Sintetizatuak:

  expr.true, expr.false: osatugabeko jauzien erreferentzia zerrenda
  expr.izena: adierazpena kalkulatuta duen aldagai edo konstantearen karaktere katea.

MURRIZTAPEN SEMANTIKOA:

Murriztapen dinamikoa da, izan ere, eragigaien balioak aztertu behar dira eta horiek ezin dira jakin exekutatu arte. Litekeena da, bestalde, exekuzio batetik bestera ere balio desberdinak izatea, baita programa zati desberdinak itzultzen aritzea ere, eta informazio hori ez dugu konpilazio garaian atributuetan, ezta sinbolo taulan ere. Salbuespen bakarra izan daiteke literalki 0 (edo 0.0) idatzita dagoenean da, baina ez dugu estatikoki egiaztatuko, ez baita ohiko kasua izango.

Nahikoa da zatitzailea zerorekin konparatuko duen agindu bat gehitzea zatiketa bakoitzaren aurretik zerorekin zatiketa inoiz gertatuko ez dela bermatzeko.

Momentuz bitarteko kodean *“goto errorea_tratatu”* jarriko dugu.

ABSTRAKZIO FUNTZIONALAK:
Kodea aldagai orokorra da eta ez da esplizitu jarriko parametroetan. Praktikaren enuntziatuan definituta dauden kodeari dagozkion metodoak ez dira hemen gehitu behar.

hasi_lista: osagaia → lista

emandako osagaia bakarrik duen lista bat sortu eta itzultzen du

SZIE:

```
mainprog →  func main ()          { kodea_hasieratu(); ag_gehitu( prog ); }
            decls { stmts } ;     { ag_gehitu( halt ); kodea_idatzi();}

stmts    →  stmt ;
            | stmts stmt ;

stmt     →  id = expr             {  ag_gehitu( id.izena || := || expr.izena ); }

            | if expr M { stmts } M
                { ag_osatu(expr.true, M1.erref); ag_osatu(expr.false, M2.erref);  }

stmt     →  id = expr     {  ag_gehitu( id.izena || := || E.izena ); }
            | if expr M { stmts } M
                { ag_osatu(expr.true, M1.erref); ag_osatu(expr.false, M2.erref);  }

expr →        expr + expr          {  -- ohiko itzulpena }
            | expr - expr          {  -- ohiko itzulpena }
            | expr * expr          {  -- ohiko itzulpena }
            | expr / expr          {  -- ohiko itzulpena }
                {  expr.izena := id_berria();
                   ag_gehitu( if ||  expr2.izena || == 0 goto errorea_tratatu);
                   ag_gehitu( expr.izena || := || expr1.izena || / || expr2.izena ); }
            | expr < expr          {  -- ohiko itzulpena }
            | expr <= expr         {  -- ohiko itzulpena }
            | expr > expr          {  -- ohiko itzulpena }
            | expr >= expr         {  -- ohiko itzulpena }
            | id                   {  -- ohiko itzulpena }
            | num_integer          {  -- ohiko itzulpena }
            | num_float            {  -- ohiko itzulpena }
```

Inplementazioa
Sarrera honetarako:

```
func main ()
{
  s3 = s3 + 6 * 6 - 4 ;
  if s2-1>s3*7 {
    s3 = s3/s2 ;
    s2 = 0 ;
  } ;
  s1 = s1 / 6 - s2 / s3 ;
}
```

honako kodea lortu nahi dugu:

```
1: prog ;
2: __t1 := 6*6 ;
3: __t2 := s3+__t1 ;
4: __t3 := __t2-4 ;
5: s3 := __t3 ;
6: __t4 := s2-1 ;
7: __t5 := s3*7 ;
8: if __t4>__t5 goto 10 ;
9: goto 14 ;
10: if s2 == 0 goto 21 ;
11: __t6 := s3/s2 ;
12: s3 := __t6 ;
13: s2 := 0 ;
14: if 6 == 0 goto 21 ;
15: __t7 := s1/6 ;
16: if s3 == 0 goto 21 ;
17: __t8 := s2/s3 ;
18: __t9 := __t7-__t8 ;
19: s1 := `_t9 ;
20: halt ;
21: write "zerorekin zatiketa gertatu da" ;
22: writeln ;
23: goto 20 ;
```

Hori lortzeko honakoak aldatu behar dira:

Atributuak:

stmts.zatizero, stmts.zatizero, expr.zatizero: zati zero egiaztapena egiteko jauzi osatugabeen erreferentzia zerrendak

SZIE:

mainprog → **func main ()** { kodea_hasieratu(); ag_gehitu(prog); }
decls **{** stmts **}** ; M
{
ag_gehitu(halt);
ag_osatu(stmts.zatizero,lortu_erref());
ag_gehitu(write "zerorekin zatiketa gertatu da");
ag_gehitu(writeln);
ag_gehitu(goto || M.erref);
kodea_idatzi();}

stmts → stmt ; { stmts.zatizero := stmt.zatizero; }
| stmts stmt ; { stmts.zatizero := bildu(stmts$_1$.zatizero,stmt.zatizero); }

```
stmt    →   id = expr                { ag_gehitu( id.izena || := || expr.izena ); }

            | if expr M { stmts } M
                 { ag_osatu(expr.true, M1.erref); ag_osatu(expr.false, M2.erref);  }

stmt    →   id = expr     { ag_gehitu( id.izena || := || E.izena ); stmt.zatizero:= expr.zatizero;}
            | if expr M { stmts } M
                 { ag_osatu(expr.true, M1.erref); ag_osatu(expr.false, M2.erref);
                   stmt.zatizero:= --- osatu itzulpena }

expr →        expr + expr
                 {  -- ohiko itzulpena
                    expr.zatizero:= bildu(expr1.zatizero,expr2.zatizero);}
            | expr - expr
                 {  -- ohiko itzulpena
                    expr.zatizero:= bildu(expr1.zatizero,expr2.zatizero);}
            | expr * expr
                 {  -- ohiko itzulpena
                    expr.zatizero:= bildu(expr1.zatizero,expr2.zatizero);}
            | expr / expr
                 {  expr.izena := id_berria();
                    expr.zatizero:= bildu(hasi_lista(lortu_erref()),
                                          bildu(expr1.zatizero,expr2.zatizero));
                    ag_gehitu( if ||  expr2.izena || == 0 goto) ; -- ez da osatzen
                    ag_gehitu( expr.izena || := || expr1.izena || / || expr2.izena ); }
            | expr < expr
                 {  -- ohiko itzulpena
                    expr.zatizero:= bildu(expr1.zatizero,expr2.zatizero);}
            | expr <= expr
                 {  -- ohiko itzulpena
                    expr.zatizero:= bildu(expr1.zatizero,expr2.zatizero);}
            | expr > expr
                 {  -- ohiko itzulpena
                    expr.zatizero:= bildu(expr1.zatizero,expr2.zatizero);}
            | expr >= expr
                 {  -- ohiko itzulpena
                    expr.zatizero:= bildu(expr1.zatizero,expr2.zatizero);}
            | id
                 {  -- ohiko itzulpena
                    expr.zatizero::= lista_hutsa(); }
            | num_integer
                 {  -- ohiko itzulpena
                    expr.zatizero::= lista_hutsa(); }
            | num_float
                 {  -- ohiko itzulpena
                    expr.zatizero::= lista_hutsa(); }
```

Osatu ezazu SZIEa (if agindua) eta inplementazioa egokitu adierazitako kodea sor dezan.