```
ANNULLABILI: statlistp , whenlistp , exprlistp

(prog)      ::== <statlist>EOF
                   statlist.next=newlabel()}<statlist>{emitlabel(statlist.next)}
(statlist)  ::== <stat><statlistp>
(statlistp) ::==;<stat><statlistp>||ε

(stat)      ::== =ID<expr>
                       |print(<exprlist>)
                       |read(ID)
                       |cond<whenlist>else<stat>
                       |while(<bexpr>)<stat>
                       |{statlist}
(whenlist)  ::== <whenitem><whenlistp>
(whenlistp) ::== <whenitem><whenlistp>|ε
(whenitem)  ::== when(<bexpr>)do<stat>
(bexpr)     ::== RELOP<expr><expr>
(expr)      ::== +(<exprlist>) | -<expr><expr>
              | *(<exprlist>) | /<expr><expr>
              | NUM|ID
(exprlist)  ::== <expr><exprlistp>
(exprlistp) ::== <expr><exprlistp>|ε
-------------------------------------------------------------------------------
------------------------------
FIRST:
ANNULLABILI: statlistp , whenlistp , exprlistp

FIRST(prog) =        FIRST(statlist)                    ={print,read,cond,while,{,=}
FIRST(statlist) =    FIRST(stat)(statlistp non conta perche stat non e
annulabile)           ={print,read,cond,while,{,=}
FIRST(statlistp) =   FIRST ;(stat)(statlistp)|ε         ={;}
FIRST(stat) =        FIRST( = ID(expr))
                          (print(<exprlist>))
                          (read(ID))
                          (cond(whenlist))(else(<stat>)) =(mi tengo solo cond per la
                          2condizione dell algoritmo )
                          (while(bexpr))
                          ({<statlist>})                 ={print,read,cond,while,{,=}

FIRST(whenlist) =    FIRST(whenitem)(whenlistp)         ={when}
FIRST(whenlistp) =   FIRST(whenitem)(whenlistp)|ε       ={when}
FIRST(whenitem) =    FIRST(when(bexpr))(do(stat))       ={when}
FIRST(bexpr) =       FIRST(RELOP(expr)(expr))           ={RELOP}
FIRST(expr) =        FIRST(+(exprlist))
                          (*(exprlist))
                          (-(expr)(expr))
                          (/(expr)(expr))
                          (NUM)(ID)
                          ={NUM,ID,+,-,*,/}
FIRST(exprlist) =    FIRST(expr)(exprlistp)             ={NUM,ID,+,-,*,/}
FIRST(exprlistp) =   FIRST(expr)(exprlistp)             ={NUM,ID,+,-,*,/}
//-----------------------------------------------------------------------------
-------------------------------------
FOLLOW:
FIRST(EOF)<=FOLLOW(STATLIST)

FOLLOW(STATLIST)<=FOLLOW(STAT)
FIRST(STATLISTP)<=FOLLOW(STAT)
FOLLOW(STATLIST)<=FOLLOW(STATLISTP)

FOLLOW(STATLISTP)<=FOLLOW(STAT)
FIRST(STATLISTP)<=FOLLOW(STAT)
//FOLLOW(STATLISTP)<=FOLLOW(STATLISTP)

FOLLOW(STAT)<=FOLLOW(EXPR)
FIRST(")")<=FOLLOW(EXPRLIST)
FIRST("else")<=FOLLOW(WHENLIST)
FIRST(")")<=FOLLOW(BEXPR)
FIRST("}")<=FOLLOW(STATLIST)

FOLLOW(WHENLIST)<=FOLLOW(WHENITEM)
FIRST(WHENLISTP)<=FOLLOW(WHENITEM)
```

```
FOLLOW(WHENLIST)<=FOLLOW(WHENLISTP)

FIRST(")")<=FOLLOW(BEXPR)
FOLLOW(WHENITEM)<=(STAT)

FOLLOW(BEXPR)<=FOLLOW(EXPR)
//FIRST(EXPR)<=FOLLOW(EXPR)

FOLLOW(")")<=FOLLOW(EXPRLIST)

FIRST(EXPRLISTP)<=FOLLOW(EXPR)
FOLLOW(EXPRLIST)<=FOLLOW(EXPR)
FOLLOW(EXPRLIST)<=FOLLOW(EXPRLISTP)

ANNULLABILI: statlistp , whenlistp , exprlistp
//-------------------------------------------------------------------------------------------
TAB:            X:

PROG            $
STATLIST        ("}" "EOF")
STATLISTP       ("}" "EOF")
STAT            FOLLOW(STATLIST)FIRST(STATLISTP)FOLLOW(STATLISTP)FOLLOW(WHENITEM) =
("}" ";" "EOF" "ELSE" "WHEN")
WHENITEM        FOLLOW(WHENLIST)FIRST(WHENLISTP)                                  =
("ELSE" "WHEN" ";")
WHENLIST        ("ELSE" "WHEN")
WHENLISTP       ("ELSE")
BEXPR           (")")
EXPR            FOLLOW(BEXPR)FOLLOW(STAT) =("}" ")" ";" "EOF" "ELSE" "WHEN")
EXPRLIST        (")")
EXPRLISTP       (")")


GUIDA:
$FOLLOW(PROG)
PROG        = FIRST(STATLIST EOF)                   ={print,read,cond,while,{,=}
STATLIST    = FIRST(<STAT><STATLISTP>)              ={print,read,cond,while,{,=}
STATLISTP   = FIRST(";" <STAT><STATLISTP>)          ={";"}
              FIRST(ε) U FOLLOW(STATLISTP)          ={"}" EOF }//
STAT        = FIRST("=" ID <EXPR>)                  ={"="}
              FIRST(PRINT (<EXPRLIST>))             ={"PRINT"}
              FIRST(READ (ID))                      ={"READ"}
              FIRST(COND <WHENLIST> ELSE <STAT>)    ={"COND"}
              FIRST(WHILE (<BEXPR>)<STAT>)          ={"WHILE"}
              FIRST({<STATLIST>})                   ={"{"}
WHENLIST    = FIRST(<WHENITEM><WHENLISTP>)          ={"WHEN"}
WHENLISTP   = FIRST(<WHENITEM><WHENLISTP>)          ={"WHEN"}
              FIRST(ε)U FOLLOW(WHENLISTP)           ={"ELSE"}
WHENITEM    = FIRST(when(<BEXPR>) do <STAT>)        ={"WHEN"}
BEXPR       = FIRST(RELOP <EXPR><EXPR>)             ={RELOP}
EXPR        = FIRST(+(<EXPRLIST>))                  ={"+"}
              FIRST(-(<EXPRLIST>))                  ={"-"}
              FIRST(* <EXPR><EXPR>)                 ={"*"}
              FIRST(/ <EXPR><EXPR>)                 ={"/"}
              FIRST(NUM)                            ={"NUM"}
              FIRST(ID)                             ={"ID"}
EXPRLIST    = FIRST(<EXPR><EXPRLISTP>)              ={NUM,ID,+,-,*,/}
EXPRLISTP   = FIRST(<EXPR><EXPRLISTP>)              ={NUM,ID,+,-,*,/}
EXPRLISTP   = FIRST(ε) U FOLLOW(EXPRLISTP)          ={")"}
-------------------------------------------------------------------------------------------
SDT OnTheFly

Prog      -> StatList       { emit(label) }
             $

StatList  -> Stat
             StatListP

StatListP -> ;
             Stat
             StatListP
```

```
138
139   Stat -> =
140                ID
141                Expr           { emit(istore) }
142
143   Stat ->      print
144               (
145                ExperList
146               )               { emit(invokestatic) }
147
148   Stat-> read
149               (
150               ID
151               )               { emit(invokestatic) }
152                               { emit(store) }
153
154   Stat ->     cond
155               WhenList
156               else
157               Stat            { emit(label) }
158
159   Stat -> while               { emit(label) }
160               (
161               Bexpr
162               )               { emit(label) }
163               Stat            { emit(GOTO) }
164                               { emit(label) }
165
166   Stat ->     {
167               StatList
168               }
169
170   WhenList -> Whenitem
171               WhenListP
172
173   WhenListP-> Whenitem
174               WhenListP
175
176   WhenListP-> EPS
177
178   When ->     when
179               (
180               Bexpr           { emit(label) }
181               )
182               do
183
184   Bexpr ->    relop
185               Expr
186               Expr            { emit(relop) }
187                               { emit(GOTO) }
188
189   Expr ->     +
190               (
191               EL
192               )               { emit(iadd) }
193
194   Expr ->     *
195               (
196               ExperList
197               )               { emit(imul) }
198
199   Expr ->     -
200               Expr
201               Expr            { emit(isub) }
202
203   Expr ->     /
204               Expr
205               Expr            { emit(idiv) }
206
207   Expr ->     NUM             { emit(ldc) }
208
209   Expr ->     ID              { emit(iload) }
210
```

```
ExperList -> Expr
             ExperListP

ExperListP -> Expr
            ExperListP      { emit(iadd) } / { emit(imul) }

ExperListP -> EPS
```