Program = P  
Declarations = DS  
Declaration = D  
Statements = SS  
Statement = S  
Variables\_List = VS  
Variable = V  
Receiver = R  
Expression = E  
Condition = C  
Type =T

Our grammar -

P->program DS ; SS end  
DS -> DD'  
D' -> epsilon  
D' -> ; DS  
D -> VS : T {VS.type = T.type; update hash table-all verables get type.}  
T -> real {T.type = real}  
T -> integer {T.type = integer}  
VS -> V{V.type=VS.type; VS’.type= VS.type}VS'   
VS' -> epsilon  
VS'L -> , V{V.type=VS’L.type; VS’R.type= VS’L.type;}VS'R  
V -> id{ id.type = v.type;  
 V’.type= V.type }V' { if(V is not used)}  
V -> int\_num  
V' -> epsilon  
V' -> [int\_num] {note check if we checked that V’.type=integer  
V’.type = Array(int\_num,value, V’.type}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
SS -> SSS'  
SS' -> epsilon  
SS' ->; SS  
S -> R = E {If R.type==E.type}  
S -> if C then S else S end if //ToCheck if we need to do something here  
S -> loop S until C end\_loop  
S -> start DS ; SS end  
R -> id{R.type = id.type; R’.type = R.type}R'  
R' -> epsilon  
R' -> {if R’.type == array}[E] {if E.type == Integer}  
E -> number {E.type = number.type}  
E -> id{ E.type = id.Type; E’.type=E.type;}E'  
E' -> epsilon  
E' -> {if E’.type == array}[E{if E.type == integer}]   
E' -> ar\_op E {if E.type == E’.type}  
C -> E rel\_op E {if EL.type == ER.type}

Original grammar -

P->program DS ; SS end  
DS -> D  
DS -> D ; DS  
D -> VS : T  
T -> real  
T -> integer  
VS -> V  
VS -> VS, V  
V -> id  
V -> [int\_num]  
SS -> S  
SS -> S ; SS  
S -> R = E  
S -> if C then S else S end if  
S -> loop S until C end\_loop  
S -> start DS ; SS end  
R -> id  
R -> id [E]  
E -> number  
E -> id  
E -> id [E]  
E -> id ar\_op E  
C -> E rel\_op E

|  |  |  |  |
| --- | --- | --- | --- |
|  | Nullable | First | Follow |
| P | - | program | EOF |
| DS | - | id int\_num | ; |
| D | - | id int\_num | ; |
| D' | + | ; | ; |
| VS | - | id int\_num | : |
| VS' | + | , | : |
| V | - | id  int\_num | , |
| V' | + | [ | , |
| T | - | real integer | ; |
| SS | - | if  loop  start  id | end |
| SS' | + | ; | end |
| S | - | if  loop  start  id | ;  until  else  end\_if  end |
| R | - | id | = |
| R' | + | [ | = |
| E | - | number id | ]  ;  rel\_op  end if  else  end  until |
| E' | + | [  ar\_op | ]  ; |
| C | - | id number | then  end\_loop |