```
E val=58
                                     T. val = 58
                                              F.val = 2
                             T. val=29
                                              digit lexval=2
                            F. val=29
                           E. val = 29
                                    T.val=1
                  E.val=28
                                    F.val=1
                  T. val= 28
                                    digit lex val= 1
                         F.val=7
                         digit. lexval=7
           Fval=4
         digit lex val = 4
5 (1) E -> E, +T {if (E, type=int) and (T, type=int)
                           then Etype = int
                           else E.type=real }
        E→T { E. type = T. type }
        T -> num num { T. type = real}
        T-> num { T type = inty
   (2) E \rightarrow E_1 + T { if (E_1, type = real \text{ and } T \text{ type} = int) then
                            E. type = real
                            grant (T. code)
                            print ('inttoreal')
                         else if (E, type=int and Ttype=real) then
                            E.type = real
                            print ('Inttoreal')
                            print (T.code)
                        else
                            E type = E. type
                            print (Toode)
                        print('+')
        E->T (E type = T type, print (T code))
        T -> num num { T type = real; T. code = num, code | 1. 1 | num, code }
        T-num { T type = int, T.code = num code }
        注:1)只能在下龙E+T向E13约时指的出Tode,因为在13约时可能需要先转换再p/int,
               不能先print 再转换
            2) 其次,E+T 向E 归约时, E已经print 过了,所以不需要再print.
 9. (1) 5 -> E { E side = right }
         E > E. := E. { if (E. side = left ) then error
                   else E, side = left E, side = right
         E→ E. + F. { if LE side = left ) then error
                   else E, side = E, side = right ( : 整个表达才吊张出现在在例)
        E > (E,) { if (E. side = loft) then error
                       E. side = right
       F→ia
        S->E { F. side = right; S. code = E code }
        E \rightarrow E_1 := E_2 { if (E side = left) then error
                    else E, side = left , E2 side = right,
                        E. code = E. code || E, code || ':= '}
        E > E, + E, { if ( E side = left) then ever
                   ele F. side = E. side = right,
                     E.code = E, code 11 E, code 11'+'4
       E > (E.) { if (E side = left) then error
                    else E. side = right,
                       E. code = E. code }
                  { E.code = id. name }
       E >id
```