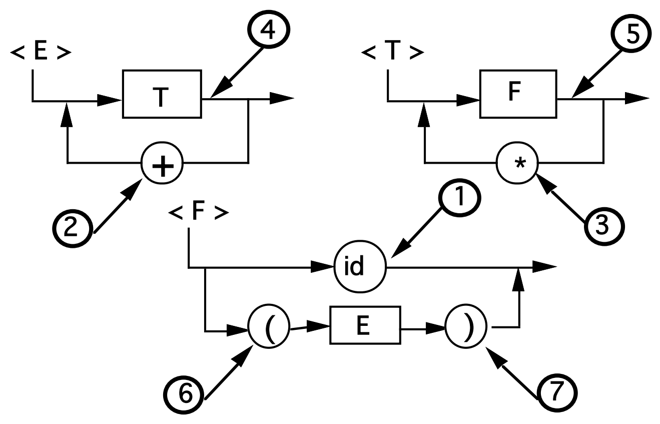
Mathematical expressions (Intermediate Representation)

Actions to produce Intermediate Representation for Math Expressions using Polish Notation with Left-Associative Operators

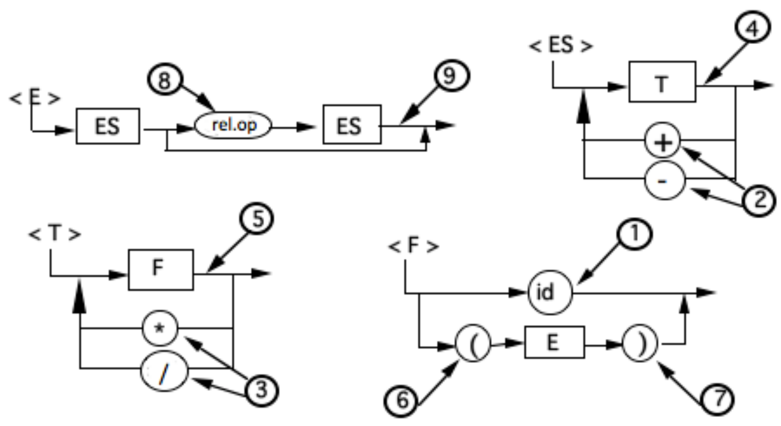
We need a Stack for "pending operators" and a Queue (VP) for output.



1.- Write the variable (id) directly into de Polish Vector (VP). /*VP.Push(id) */
2.- Push (+) into Operator's Stack (POper)
3.- Push (*) into Operator's Stack (POper)
4.- If POper.top = '+' then:
VP.Push(POper.pop()
5.- If POper.top = '*' then:
VP.Push(POper.pop()
6.- POper.Push(False-bottom mark)
7.- POper.Pop(False-bottom mark)

Actions to produce Intermediate Representation for Math Expressions using Quadruples with Left-Associative Operators, including basic semantic.

We need a Stack for "pending operators" (POper), a Stack for "pending operands" (PilaO), a Stack for corresponding types (PTypes and a Queue (Quad) for output.



```
1.  PilaO.Push(id.name) and
    PTypes.Push(id.type)
2.  POper.Push(+ or -)
3.  POper.Push(* or / )
4.  If POper.top() == '+' or '-' then
    right_operand= PilaO.Pop()
    right_Type=PTypes.Pop()
    left_operand=PilaO.Pop()
    let_Type=PTypes.Pop()
    operator= POper.Pop()
    result_Type= Semantics[left_Type,
                     right_Type, operator]
4…continue…
        if (result_Type != ERROR)
                result ←AVAIL.next()
                generate  quad= (operator, left_operand, right_operand, result)
                Quad.Push(quad)
                PilaO.Push(result)   PTypes.Push(result_Type)
                If any operand were a temporal space, return it to AVAIL
        Else
                ERROR ("Type mismatch")
5.      If POper.top() == '*' or '/' then
                = to #4 with *,/
6.      POper.Push(False bottom mark)
7.      POer.Pop(False bottom mark)
8.      POper.Push(rel.op)
9.      If POper.top() == rel.op then
                = to #4 with >, <,..
```