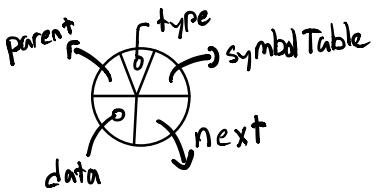
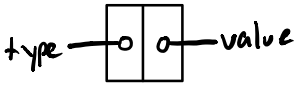
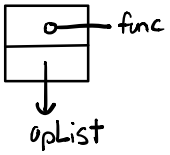
AST_NODE



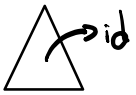
AST_NUMBER



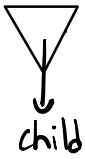
AST_FUNCTION



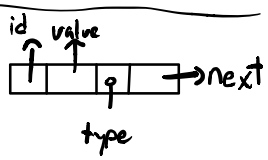
AST_SYMBOL



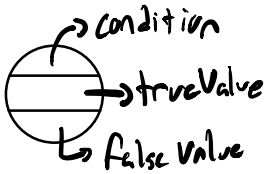
AST_SCOPE

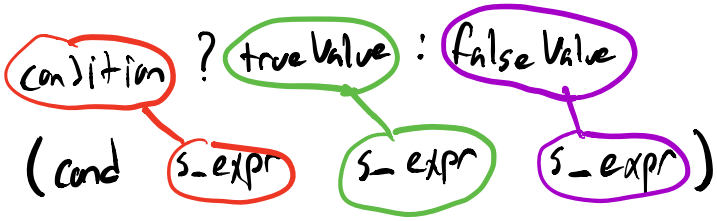


SYMBOL_TABLE_NODE

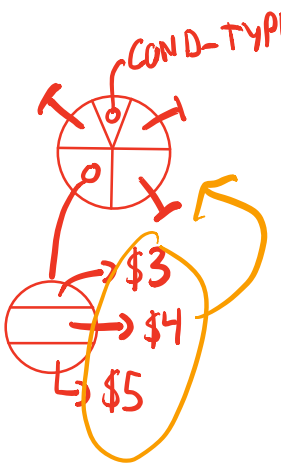


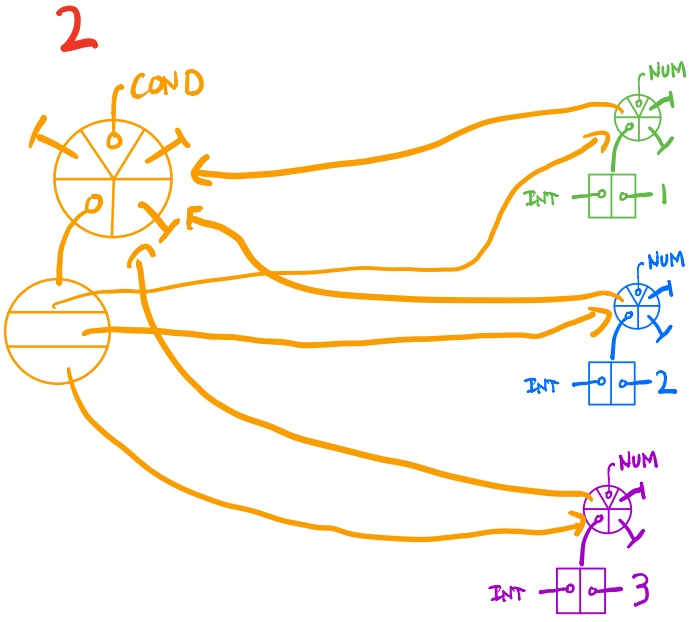
AST_CONDITIONAL





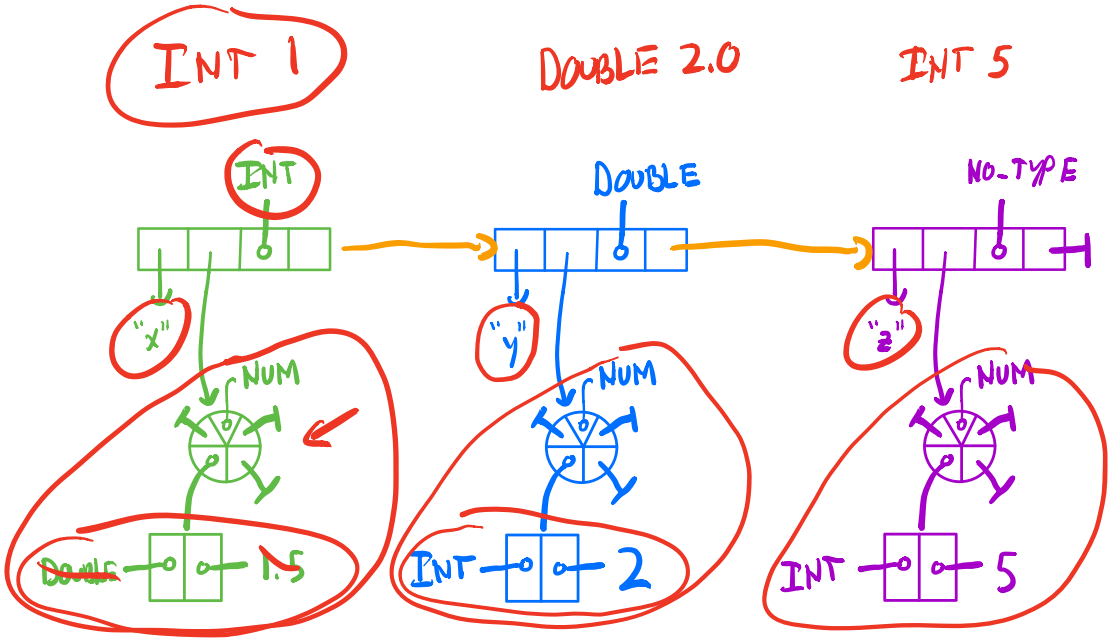
S_expr ::= (COND s_expr s_expr s_expr)
$1 $2 $3 $4 $5 $6





"int"
"double"

(cond 1 2 3)



((let (int x 1.5) (double y 2) (z 5)) (add x y z))

let-elem ::= (symbol s-expr)

| (TYPE SYMBOL s-expr)

rand
read
print

(print 1)

((let (x 1)) (print x))

((let (x (rand))) (add x x))

0.27

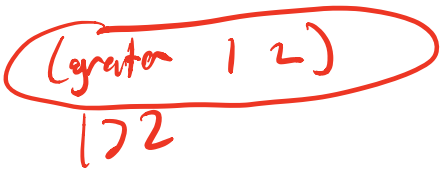
less

greater

equal

(less 1 2)

1 < 2



1 ≯ 2