## Code Specification

|  |  |
| --- | --- |
| Functions | Code Templates |
| run⟦program⟧ | run⟦program → definition\*⟧ =  # SOURCE {file}  define[[definition\*i]] |
| value⟦expression⟧ | value⟦variable:expression → name:string⟧ =  Addresss[variable]  value⟦intLiteral:expression → intValue:int⟧ =  value ⟦floatLiteral:expression → floatValue:float⟧ =  value ⟦charLiteral:expression → charValue:char⟧ =  value ⟦functionCallExpression:expression → name:string expression\*⟧ =  value ⟦structAccess:expression → expression name:string⟧ =  value ⟦arrayAccess:expression → left:expression right:expression⟧ =  value ⟦cast:expression → castType:type expression⟧ =  Value[[value]]  IF castType == intType && expression.type == charType   b2i  IF castType == intType && expresión.tipó == realType   f2i  IF castType == charType && expresión.tipó == intType   i2b  IF castType == realType && expresión.tipó == intType   i2f  value ⟦arithmetic:expression → left:expression operator:string right:expression⟧ =  Value[[left]]  Value[[right]]  IF operator == ‘+’  ADD  IF operator == ‘-‘  SUB  IF operator == ‘\*’  MUL  IF operator == ‘/’  DIV  value ⟦comparison:expression → left:expression operator:string right:expression⟧ =   value ⟦logicBinary:expression → left:expression operator:string right:expression⟧ =  Value[[left]]  Value[[right]]  IF operator == ‘<’  LT  IF operator == ‘>‘  GT  IF operator == ‘<=’  LE  IF operator == ‘>=’  GE  IF operator == ‘==’  EQ  IF operator == ‘!=’  NE  Value[[left]]  Value[[right]]  IF operator == ‘&&’  AND  IF operator == ‘||‘  OR  value ⟦logicUnary:expression → operator:string expression⟧ =  Value[[expresssion]]  IF operator == ‘!’  NOT |
| define⟦definition⟧ | define⟦varDefinition:definition → name:string type⟧ =  # GLOBAL {end.line}  define⟦structDefinition:definition → name:string fieldDefinition\*⟧ =  define⟦functionDefinition:definition → name:string varDefinition\* type? definition\* statement\*⟧ =  # LINE {end.line}  value[[expression]  OUT  define⟦fieldDefinition:definition → name:string type⟧ = |
| execute⟦statement⟧ | execute⟦print:statement → expression⟧ =  # LINE {end.line}  value[[expression]  OUT  execute⟦read:statement → expression⟧ =   execute⟦functionCallStatement:statement → name:string expression\*⟧ =   execute⟦assignment:statement → left:expression right:expression⟧ =   execute⟦conditional:statement → expression ifStatements:statement\* elseStatements:statement\*⟧ =   execute⟦while:statement → expression loopStatements:statement\*⟧ =   execute⟦return:statement → expression?⟧ =  #LINE {end.line}  Address[[left]]  Value[[right]]  STORE  CALL {name}  IF funcionLlamada.definicion.tipo ¡= null  POP  Address[[read]]  IN  STORE |
| address⟦expression⟧ | address⟦ident:expression → name:string ⟧ =  address ⟦struct: expression → left:expression right:expression⟧ =  address ⟦array: expression → left:expression right:expression⟧ = |