A brief description

Abstract Syntax Tree

The class EvalVisitor is a visitor class that extends CCALBaseVisitor<Integer>. It is designed to traverse the AST generated by the ANTLR parser for CCAL. The visitor pattern is a design pattern that separates the algorithm from the object structure on which it operates. In the context of the AST, a visitor is an object that traverses the tree and performs actions at each node.

SymbolTable

ST (Symbol Table): It's represented as a Map<String, Symbol>, where each symbol (variable, function, etc.) is associated with a unique identifier (key) and its corresponding attributes (value).
Stack (stack): A stack is used to manage scopes. The enterScope method pushes a special marker ("#") onto the stack when entering a local scope. The exitScope method pops symbols from the stack until the marker is encountered, effectively removing local declarations.
addSymbol Method: Adds a symbol to the symbol table and pushes it onto the stack. This method is used when a new variable or function is declared.
getSymbol Method: Retrieves a symbol from the symbol table based on its identifier.
getType Method: Retrieves the type of a symbol given its identifier.
compareType Method: Compares the type of a symbol with a specified type.
checkConstError Method: Checks if a constant can be updated (an error is thrown if it cannot be).
compareTypeValue Method: Compares the type of a declaration to the type of a value being assigned and throws an error if they do not match

Semantic Checks

Constant Update Check (checkConstError method):
Ensures that constants cannot be updated after they are declared. If an attempt is made to update a constant, a RuntimeException is thrown, advising the use of a variable instead.
Type-Value Compatibility Check (compareTypeValue method):
Compares the type of a constant declaration to the type of its assigned value. If they do not match, a RuntimeException is thrown, indicating a mismatch between the declared type and the assigned value type.
Void Variable Declaration Check (visitVar_decl method in EvalVisitor class):
Ensures that variables cannot be declared with the type "void." If a variable with the type "void" is encountered, a RuntimeException is thrown.
Undefined Variable Check (visitFragm method in EvalVisitor class):
Checks if an identifier (variable or constant) is used before being declared. If an undefined identifier is encountered, a RuntimeException is thrown.
Function Call Argument Count Check (visitFunctionCallStatement and visitFunctionCallOp methods in EvalVisitor class):
Verifies that the number of arguments in a function call matches the number of parameters in the function definition. If the counts do not match, a RuntimeException is thrown.
Function Declaration Check (visitFunction method in EvalVisitor class):
Adds a function to the symbol table and checks for duplicates. If a function with the same name already exists, a RuntimeException is thrown.

Generating an Intermediate Representation using 3-address code

The IrVisitor class in my code serves as a visitor for the abstract syntax tree (AST) generated by ANTLR for the CCAL language. It extends CCALBaseVisitor<String> and is used to produce intermediate representation (IR) code. The class initializes an output file handler, a branch counter, a flag to track whether it's inside a branch, and a string to store the current block of code. The methods in this class handle specific AST node types, such as assignment statements, fragments, while statements, and if-else statements.

https://medium.com/@dinis.cruz/ast-abstract-syntax-tree-538aa146c53b
https://en.wikipedia.org/wiki/Three-address_code