Andrew Plum

Compilers

5/15/2024

Notes

the DeclNode, StmtNode, and ExpNode in your abstract syntax tree (AST) likely represent the following components of your programming language:

1. **DeclNode**:  
   DeclNode represents declarations in your language, such as variable declarations, function declarations, or type declarations. Declarations introduce new identifiers (names) and associate them with a particular type or storage location. Examples of declarations include:

* Variable declarations (e.g., int x;, bool flag;)
* Function declarations (e.g., int add(int a, int b);)
* Type declarations (e.g., typedef int myInt;)

1. **StmtNode**:  
   StmtNode represents statements in your language, which are the basic units of execution. Statements perform actions or control the flow of execution. Examples of statements include:

* Assignment statements (e.g., x = 5;)
* Control flow statements (e.g., if (condition) {...} else {...}, while (condition) {...})
* Loop statements (e.g., for (int i = 0; i < 10; i++) {...})
* Function call statements (e.g., result = add(2, 3);)
* Return statements (e.g., return x;)

1. **ExpNode**:  
   ExpNode represents expressions in your language, which are combinations of values, operators, and function calls that evaluate to a single value. Expressions can be simple or complex, and they can be used in various contexts, such as assignments, control flow conditions, or function arguments. Examples of expressions include:

* Arithmetic expressions (e.g., x + y \* z, a / b)
* Logical expressions (e.g., a && b, !c)
* Relational expressions (e.g., x < y, a == b)
* Function calls (e.g., sqrt(x), max(a, b))
* Literal values (e.g., 42, 3.14, 'c', "hello")

These three types of nodes (DeclNode, StmtNode, and ExpNode) are common in ASTs for programming languages, as they represent the fundamental building blocks of most languages: declarations, statements, and expressions.During the parsing and semantic analysis phases of your compiler, these nodes will be constructed and organized into a tree-like structure that represents the syntactic and semantic structure of the input program. This AST can then be used for further analysis, optimization, or code generation.