CS1319: PLDI - Assignment 4

Hrsh Venket & Santripta Sharma

October 2023

1 Augmenting the Grammar

We begin our efforts on the semantic analyser/TAC generator for nanoC by augmenting the grammar to add attributes to existing symbols, and add new dummy/marker symbols, to help us generate the final code. Towards this end:

- We first define our attributed grammar, specifying which attribute is associated with each terminal/non-terminal symbol.
  - All constants (integer, char, string literals) have their attributes as their literal values. The flex lexer returns these literal values to the bison parser.
  - IDENT terminals have the identifier string as their attribute.
  - All expressions are attributed with a struct, containing a pointer to a symbol table entry, a pointer to a true/false list, and a pointer to a next list. Depending on the type of expression (what path it takes in the parse tree), the pointers to the true/false and next lists may be null.
  - All declaration-related non-terminals are attributed with the symbol struct, which they incrementally construct, filling in details through the reductions. Once the declaration is completed, these are appended into the symbol table, and any initialisation code is emitted. An optional (nullable) attribute tracking an associated initialiser expr (type a = expr;) is also available.