Safi Nassar - assignment p4

Usage:

- `make clean && make testclean && make`
- `make test`

Executing the test will generate output file of the formated program.

Prerequisites & dependencies:

- OpenJDK 11
- Java Cup
- JLex

Tests documentation:

Name analyzer for C– programs:

- Programs are represented as ASTs.
- Name analyzer functionality:
    1. Build scope symbol tables (using SymTable class)
    2. Detect static semantic name declaration
    3. Detect static semantic usage errors & Add AST IdNode links to corresponding symbol-table entry: for nodes representing a use of name.

C– names & scoping rules:

- [x] Allows same name declarations in non-overlapping/nested scopes.
- [x] Uses of undeclared names - names must be declared before they are used.

variable declaration

- [x] Duplicate/Multiply declared names
- Bad declaration:
    - [x] type void declaration for non function
    - [x] variable of bad struct type - i.e. the name of the struct type doesn't exist or is not struct type.
- [x] If you find a bad variable declaration (a variable of type void or of a bad struct type), give an error message and add nothing to the symbol table.

struct

- struct declaration is scope of one level above it's body.
- [x] duplicate variable/function name to struct: do not add symbol entry, do not process variables of the struct.
- [x] Duplicate field: ignore second occurrence.
- [x] If a struct is used without declaration like a.b, then you can report two errors (undeclared ID and dot access of non-struct type) or you can just report undeclared ID.
- struct handling issues:
  - Bad struct access: name analysis `LHS.RHS` should check:
    - [x] left-hand side of the dot-access is not a name already declared to be of a struct type
    - [x] right-hand side of the dot-access is not the name of a field for the appropriate type of struct.
  - struct definition:
    - [x] name of the struct type can't be a name that has already been declared.
    - [x] A variable x inside a struct with the same name as another variable inside the struct is illegal (fields of a struct must be unique to that particular struct).
    - [x] A variable inside a struct with the same name as a variable or a function outside the struct is legal.
  - [x] struct declaration: on variable declaration that is a struct - check struct type has been previously declared and is actually the name of a struct type.

function

- [x] Formal parameters & function body are on same scope.
- [x] Illegal same name to another function or variable in same scope - do not add entry for second occurence, but continue processing formals and body of the functions.
- [x] if formal paramter = body variable declared: add formal and ignore param.
- [x] Duplicate formal/local: create SymTable, add first parameter/local variable, report the error and then continue with processing.

if/else/while

- [x] if/else and while statements have their own scope. So, names can be reused inside these statements.
- [x] The if part and the else part have different scopes. So, the same name can be declared in both of them.

Tasks

- [x] Modify the TSym class (by including some new fields and methods and/or by defining some subclasses).
- [x] Modify the IdNode class in ast.java (by including a new TSym field and by modifying its unparse method). Free to make any changes to ast.java
- [x] Modify P4.java (an extension of P3.java).
- [x] Modify the ErrMsg class.
- [x] Write two test inputs: nameErrors.cminusminus and test.cminusminus to test your new code.
- [x] name analyzer will use symbol table's globalLookup method to set IdNode's symbol table link.
- [x] name analyzer implemented by writing appropriate methods for the different subclasses of ASTnode. Exactly what methods you write is up to you (as long as they do name analysis as specified).
  - [x] Some nodes's methods need to add a new hashtable (scope enter) or remove hashtable (scope exit) from symbol table.
  - [x] Some node's methods will process declarations - checking for name errors, and adding appropriate symbol table enteries.
  - [x] Some node's methods process statements in the program - checking declarations for each used name, and adding links for used names IdNodes of those that were declared.
- [x] Handling errors: Your compiler should quit after the name analyzer has finished if any errors have been detected so far (either by the scanner/parser or the name analyzer). To accomplish this, you can add a static boolean field to the ErrMsg class that is initialized to false and is set to true if the fatal method is ever called (warnings should not change the value of this field). Your main program can check the value of this field and only call the unparser if it is false.
- [x] Traversal of AST should be similar to unparsing traversal implemented using `unparse` method. Only involved nodes should be traversed, and their parent classes should have an abstract methods to implement on all their subclasses (e.g. TypeNode subclasses should not be).
  - [x] declare an abstract name-analysis method for the DeclNode class, because the method for the DeclListNode class will call that method for each node in the list.

Submission:

- [x] Create pdf from markdown: `pandoc README.md -o <lastname.firstname.Pn.pdf>`
  - [ ] generate markdown from javadoc and remove redundant comments or
  - [ ] generate javadoc to extract method headers: `find . -type f -name "*.java" | xargs javadoc -d ../javadoc`
- [ ] Add headers for each file
- [x] Verify code format

- [ ] Verify code execution on CSL machines
- [x] lastname.firstname.lastname.firstname.P4.zip +—+ deps/ +—+ ast.java +—+ cminusminus.cup +—+ cminusminus.jlex +—+ DuplicateSymException.java +—+ EmptySymTableException.java +—+ ErrMsg.java +—+ Makefile +—+ P4.java +—+ TSym.java +—+ SymTable.java +—+ nameErrors.cminusminus +—+ test.cminusminus +—+ lastname.firstname.lastname.firstname.P4.pdf