README for the symbol-table project

Wuu Yang
National Chiao-Tung University, Taiwan, Republic of China
July 22, 2020

Three tasks in this project are

1. building AST.
2. building Symbol table
3. checking declarations and types

In the 3rd project, you need to build an abstract syntax tree (AST) during parsing. The file ast-node-for-project.pdf contains sample AST nodes. You should show me your abstract syntax tree on the computer screen. You can implement a function to traverse AST and print the name/kind of the each node during the traversal.

For the 3rd part of the project, you also need to write code to process declarations and to build symbol tables. You code must print a message in the following occassions:

1. every time a new symbol is entered into the symbol table
2. every time a new scope is generated
3. every time a scope is closed
4. every time when a symbol table is created
5. every time when a symbol table is destroyed
6. dump a symbol table when the symbol table is destroyed
7. dump all symbol tables at the end of the program

You need to implement a function `printSymbolTable()`, which will print a symbol table.

For each entry in the symbol table, you need to add information regarding kind (such as variable name, function name, type name, etc.), type (such integer, real, etc.), scope, address (if available), etc.

For this 3rd project, you may assume the parser always succeeds. However, there could be errors in declarations, expressions, and statements. The compiler needs to perform type checking and print a message for each error. A sample error message is

```
Line 35:  variables a and b should have the same type.
```

The directory SymtabTestCases contains several test cases. Your program probably cannot find all errors in the test cases. However, your program should try to find as many errors as possible.

Difficult Issues

The rules for checking types are stated in the file 01-minipascal-spec.pdf. You compiler should check semantic errors. Common semantic errors include

1. Declare two variables with identical names in two scopes. Can you find the appropriate variable when that name is used in an expression?

2. Declare two variables with identical names in the same scope. Can you catch the duplicate declarations?

3. Declare a variable and a function with identical names in the same scope. Can you catch the duplicate declarations?

4. Declare a variable and a function with identical names in the differnt scopes. Can you find the appropriate usage when that name is used?

5. undeclared variables

6. undeclared types (not in this mini-pascal project)
7. undeclared functions (not in this mini-pascal project)
8. undeclared classes (not in this mini-pascal project)
9. arguments' types and numbers are wrong.
10. type errors in arithmetic expressions
11. redeclared variables
12. redeclared types (not in this mini-pascal project)
13. redeclared functions
14. redeclared classes (not in this mini-pascal project)
15. cycles in inheritance hierarchy (not in this mini-pascal project)
16. cycles in structure containment (not in this mini-pascal project)

Tools and ParserTestCases

There are no special tools for this symbol-table project. You need to write C/C++ code yourself.

Handin

For the 3rd (ast, symbol table, and type checking) project, you need to turn in your lex files, yacc files, semantic routine files, test cases, and the executable code. You need to prepare for Makefile. The TA will run your parser with

```
make run
```

You also need to write a readme.txt file for the project, telling the TA how to run your program and additional details.

Put all of the above files in a single zip file which will be named "DDDDDDD-symtab.zip", where DDDDDDD is your student id. Upload the zip to the new e3 platform.

Deadline of the 3rd project is December 10, 2020, Thursday, 23:55 pm.