SOFTWARE DESIGN DOCUMENT

MINI-PASCAL COMPILER

Version 0.3

William Mork

Augsburg University

PROJECT OVERVIEW

Introduction

This project, written in Java 8, is a compiler which parses Mini-Pascal code to generate MIPS assembly code. Please refer to the “Project Structure” section for clarity on the files and modules used in the project.

Module 1: Scanner

The scanner module reads a Mini-Pascal text file and scans each line. Keywords and symbols which are recognized as valid (listed below) by the scanner are converted into “tokens”, which will later be handled by the parser module.

Scanner.java is a file which has been generated by JFlex, a lexical analyzer (scanner) generator. The generator uses a specified set of token types, expected patterns, and lexical rules to create a deterministic finite automata (DFA) which is used to construct the aforementioned token stream.

Token.java defines a token object containing the token lexeme and type.

TokenType.java enumerates the list of valid keywords and symbols.

Valid keywords:

AND ARRAY BEGIN DIV DO ELSE END FUNCTION IF INTEGER MOD NOT OF OR PROCEDURE PROGRAM REAL THEN VAR WHILE READ WRITE RETURN

Valid symbols (token type is listed first, followed by the symbol itself):

SEMI ; COMMA , PERIOD . COLON : LBRACE [RBRACE] LPAREN (RPAREN) PLUS + MINUS - EQUAL = NOTEQ <> LTHAN < LTHANEQ <= GTHAN > GTHANEQ >= ASTERISK * FSLASH / ASSIGN :=

Module 2: Parser

The recognizer / parser module employs an instance of the scanner class to iterate through tokens of an input stream and match them with the production rules articulated in the MicroPascal grammar (Grammar.pdf).

To use the current version of the recognizer class, create an instance and then call the top-level function, exp(). If the function returns without encountering an error, the file is an acceptable MicroPascal program or contains an acceptable expression.

The Recognizer class can be used to read the token stream of a pascal file or a provided input String. As of version 0.3, the Recognizer class is not a true parser. In later builds of this project, the Recognizer will be developed towards a full recursive-descent parser and will construct the nodes of a syntax tree.

Module 3: Symbol Table

The symbol table module is used to store information on the identifiers used in a pascal program. As of version 0.3, the symbol table is unable to handle subprograms, such as functions and procedures. In later builds of this project, the symbol table will manage multiple hashmaps, stored in a stack, with the global scope (program) being pushed onto the stack first.

SymbolTable.java contains a single hashmap, whereby each key-value pair contains the lexeme of an identifier (key) as well as information appropriate to the “kind” of identifier recognized by the parser. A full list of the symbols contained in the hashmap can be generated by calling the toString() method of this class.

Symbol.java defines a symbol object, with discrete constructors for every “kind” of identifier that the parser might encounter. A The toString() method of this class can be called

Kind.java enumerates the list of valid identifiers that can be added to the symbol table.

Project Structure

- MorkCompiler
  - documentation
    - Grammar.pdf
    - SDD.pdf
    - User Manual.pdf
  - src
    - pascal
      - simple.pas
      - simplest.pas
    - scanner
      - LookupTable.java
      - Scanner.flex
      - Scanner.java
      - Token.java
      - TokenType.java
      - ScannerTest.java
    - parser
      - Recognizer.java
      - RecognizerTest.java
    - symbol table
      - SymbolTable.java
      - Symbol.java
      - Kind.java
      - SymbolTableTest.java
  - .gitignore
  - README.md

Master Changelog

Commit ID	Commit Tag	Version	Description	Date
Imported old files.				
99965c6	Import	0.0		3/3/2019
Finished Scanner edits for first module.				
1447a47	N/A	0.0		3/3/2019
Final commit for Module 1.				
c3518ff	Scanner	0.1	Finished Scanner module and related unit testing.	3/3/2019
Imported example files.				
eb79317	N/A	0.1		3/3/2019
Finished Recognizer functions.				
3690c92	N/A	0.1		3/3/2019
Final commit for Module 2.				
5c2887a	Recognizer	0.2	Finished Recognizer module and related unit testing.	3/3/2019
Fixed readability issues with version control history.				
75fc787	N/A	0.2	Old Master branch deleted and restored.	3/3/2019
Final commit for Module 3. (Updated SDD)				
N/A	SymbolTable	0.3	Finished Symbol table module and related unit testing.	3/12/2019