2019

DorseyCompiler

A COMPILER MADE BY COHL DORSEY

COHL DORSEY

Overview

This compiler is built to compile a very minimalistic version of pascal into MIPS assembly. The compiler will take in a file containing the code for this mini version of pascal where it will then scan the file using the rules defined in the scanner class of the java code. It will then be parsed by the parser class and compiled into a functioning MIPS assembly program. This assembly program will then be able to be run and function as defined by what was written in the pascal code.

The Scanner

This scanner is programmed in java and is generated using jflex. The scanner will take in a file and scan it, once the file has been scanned it will print out to the terminal a list of accepted tokens and unexpected tokens. The code for the Scanner and all its needed classes are stored in the dorseyScanner package.

JFlex File

The jflex file holds the tags to generate the java scanner file. It also contains the grammar and definitions for how each grammar object should be treated. When compiled this creates the the Scanner.java file that can then be compiled.

Scanner

This Scanner is the java file created from the jflex file. The scanner will take in a file and scan it for the tokens that have been defined. To see what these tokens are you can find the list of them in the key words and symbols section in this document. Once the file has finished being scanned the scanner will output the tokens that were defined and it will also show the ones that are errors.

TokenType

This class contains all the types of objects that a token can be. When the class is called in the driver it has a list of types that it iterates through. The correct token type is then taken and passed into the scanner.

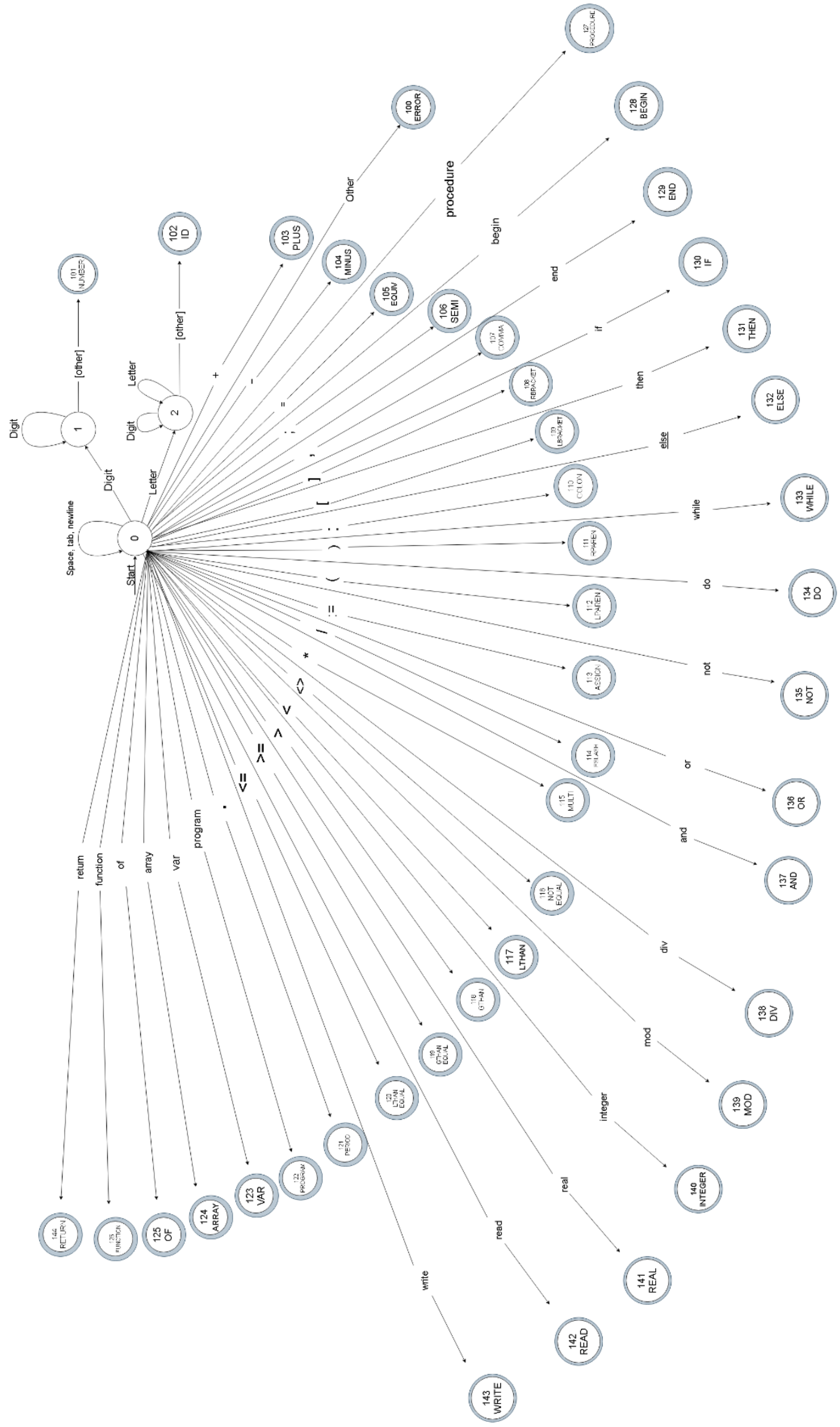
Token

This class is the constructor for a Token object. This class creates a Token that once its type is determined gets used by the scanner to parse the file that was given as input.

MyScanner

This is the main class of the scanner. It takes in a file to be scanned, creates a scanner object. The file is then passed into that scanner where it generates the tokens and processes them according to the definitions that are defined by its type.

The DFS



FSM Design: Designed by Marissa Allen and Cohl Dorsey

List of Keywords/Reserved Words:

array

program

var

do

if

else

then

of

function

procedure

begin

end

while

and

or

not

div

mod

integer

real

List of Symbols:

;

,

[

]

)

(

{

}

:

+

-

*

=

<>

<

<=

>=

>

/

:=

.

The Parser

The Parser is written in all Java code and has test files written using Junit. The Parser is responsible for using the scanner to parse through the pascal code and break the code up into the appropriate symbolic tokens. The parser package consists of several class files that work in conjunction such as the actual Parser class, and SymbolTable class which creates a table of the symbols. The parser package contains several methods that help construct the

Parser

The Parser uses the scanner package to parse through the pascal code and tokenize it into its specified tokens. It then adds any new symbols into the token table.

SymbolTable

The SymbolTable class is the class used to represent the symbol table that stores the tokens. It includes methods to see if a token already exists in the table, it also has methods to determine what type of token is being stored into it based on the grammar that we specified in the Recognizer.

Recognizer

The Recognizer class contains our actual pascal grammar. When this class goes over the code it turns the code into the appropriate token based on what is specified in our grammar.

CompilerMain

This class represents the main aspect of the compiler. When the compiler is fully complete this is the class that will manage the compiler as a whole.

Change Log

02/15/2019	Added the Recognizer section of the compiler.
03/9/2019	Added the Symbol Table section of the compiler.
03/14/2019	Added the CompilerMain section of the compiler.