# San José State University Department of Computer Science

# CS/SE 153 Concepts of Compiler Design

Fall 2023 Instructor: Ron Mak

# **Assignment #2**

Assigned: Monday, August 28

Due: Monday, September 11 at 4:00 PM

Team assignment, 100 points max

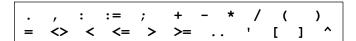
#### Pascal scanner

The purpose of this assignment is to give you practice writing a scanner for Pascal.

Start with the **Scanner** and **Token** classes in <u>Simple.zip</u> that we went over in class. Modify the classes to handle the following Pascal reserved word tokens:

PROGRAM BEGIN END REPEAT UNTIL WRITE WRITELN DIV MOD AND OR NOT CONST TYPE VAR PROCEDURE FUNCTION WHILE DO FOR TO DOWNTO IF THEN ELSE CASE OF

Handle the following Pascal special symbol tokens:



Also recognize these tokens:

IDENTIFIER INTEGER REAL CHARACTER STRING END\_OF\_FILE ERROR

You can make any modifications that you deem necessary to the other classes. For a more complete list of Pascal tokens, see the syntax diagrams at <a href="http://primepuzzle.com/tp2/syntax-diagrams.html">http://primepuzzle.com/tp2/syntax-diagrams.html</a>

#### Comments

Your scanner should treat each comment as it would treat a blank – comments should be ignored. Pascal comments are enclosed in curly braces { and }.

### Strings and character literals

A literal Pascal string is enclosed in single quotes. If a single quote character is part of a string, it is represented by two consecutive single quotes. For example, 'It''s' contains the characters It's. It is possible to have the empty string: ''

A literal Pascal character is simply a string with only one character. For example: 'a'

#### **Test files**

Test your code on test input file SquareRootTable.pas:

```
PROGRAM SquareRootTable;
VAR
   whole, frac : integer;
   number : real;
FUNCTION sqroot(n: real) : real;
   VAR
        root, prev, diff : real;
   BEGIN
       root := n;
       prev := root;
        REPEAT
            root := (n/root + root)/2;
            diff := prev - root;
            prev := root;
        UNTIL diff < 0.0000001;
        sqroot := root;
   END;
BEGIN
   writeln('Square Root Table');
   writeln;
   write('
               ');
   FOR frac := 0 TO 9 DO BEGIN
       write(frac/10.0:10:1);
   END;
   writeln;
    FOR whole := 1 TO 25 DO BEGIN
        write(whole:5);
        FOR frac := 0 TO 9 DO BEGIN
            number := whole + frac/10.0;
            write(sqroot(number):10:6);
        END;
        writeln;
   END:
END.
```

Test input file <a href="ScannerTest.txt">ScannerTest.txt</a> will give your scanner and token classes a good workout:

```
{This is a comment.}
{This is a comment
that spans several
 source lines.}
Two{comments in}{a row} here
{Word tokens}
Hello world
begin BEGIN Begin BeGiN begins
{String tokens}
'Hello, world.'
'It''s Friday!'
1 1
'A' 'x' '''
'This string
spans
source lines.'
{Special symbol tokens}
+ - * / := . , ; : = <> < <= >= > ( ) [ ] { } } ^ ...
+-:=<>=<=....
{Number tokens}
0 1 20 000000000000000032 31415926
3.1415926 3.1415926535897932384626433 .14
{Bad tokens}
3.14.15926
What?
'String ''not'' closed
```

## **Expected output**

Your output for input file ScannerTest.txt should be similar to the following:

```
Tokens:
   IDENTIFIER : Two
   IDENTIFIER : here
   IDENTIFIER : Hello
   IDENTIFIER : world
        BEGIN : begin
        BEGIN : BEGIN
        BEGIN : Begin
        BEGIN : BeGiN
   IDENTIFIER : begins
       STRING : 'Hello, world.'
        STRING : 'It's Friday!'
        STRING : ''
    CHARACTER : 'A'
    CHARACTER : 'x'
    CHARACTER : '''
       STRING : ' ' '
        STRING : ''''
       STRING : 'This string
spans
source lines.'
         PLUS : +
        MINUS : -
         STAR : *
        SLASH : /
 COLON EQUALS : :=
       PERIOD : .
        COMMA:,
    SEMICOLON : ;
        COLON : :
       EQUALS : =
   NOT EQUALS : <>
    LESS THAN : <
  LESS EQUALS : <=
GREATER EQUALS : >=
 GREATER THAN : >
       LPAREN : (
       RPAREN : )
     LBRACKET : [
     RBRACKET : ]
TOKEN ERROR at line 24: Invalid token at '}'
        ERROR : }
        CARAT : ^
      DOT DOT : ..
         PLUS : +
        MINUS : -
 COLON EQUALS : :=
   NOT EQUALS : <>
       EQUALS : =
  LESS_EQUALS : <=
       EQUALS : =
      DOT DOT : ..
      DOT DOT : ..
       PERIOD : .
      INTEGER: 0
      INTEGER: 1
      INTEGER: 20
      INTEGER: 0000000000000000032
      INTEGER: 31415926
         REAL : 3.1415926
         REAL : 3.1415926535897932384626433
       PERIOD : .
      INTEGER: 14
TOKEN ERROR at line 32: Invalid number at '3.14.15926'
        ERROR : 3.14.15926
   IDENTIFIER : What
TOKEN ERROR at line 33: Invalid token at '?'
        ERROR : ?
TOKEN ERROR at line 34: String not closed at ''String 'not' closed'
       STRING : 'String 'not' closed
```

### What to submit to Canvas

- A new version of SimpleJava.zip that includes your modified Scanner and Token classes.
- Text files of output from running your scanner on input files SquareRootTable.pas and ScannerTest.txt.

Submit to Assignment #2: Pascal Scanner

There should be only one submission per team.

#### **Rubric**

Your submission will be graded according to these criteria:

Criteria	Maximum points
Reserved words handled properly.	30
Special symbols handled properly.	30
Token errors handled properly.	30
Good output format.	10