San José State University
Department of Computer Science

CS 153
Concepts of Compiler Design

Section 1
Fall 2020
Instructor: Ron Mak

Assignment #3

Assigned: Thursday, September 3
Due: Thursday, September 17 at 8:30 AM
Team assignment, 100 points max

Simple Pascal interpreter

The purpose of this assignment is to give you practice writing a parser and an interpreter for simplified Pascal. You can start this assignment with your solution to Assignment #2, or you can use Asgn02Java.zip.

Modify the frontend parser to include the **`WHILE`**, **`FOR`**, **`IF`**, and **`CASE`** statements of the simplified Pascal. Use the syntax diagrams and the parse trees shown in the lecture notes. Then modify the interpreter's backend executor to execute those statements.

Make any necessary modifications to the other classes. For example, you may need to add more parse tree node types.

Suggested order

Here is a suggested order to implement the additional statements:

1. **`WHILE`** statement. It's similar to the **`REPEAT`** statement.
2. **`IF`** statement. Handle statements with and without an **`ELSE`** part. Be sure to handle a "dangling else" correctly.
3. **`FOR`** statement. It's a more elaborate parse tree, so build it carefully according to the parse tree in the lecture notes.
4. **`CASE`** statement. The most elaborate parse tree. Build it very carefully according to the parse tree in the lecture notes.

For each statement, first make sure your parse tree is correct by visually inspecting the tree that the **`-parse`** option prints. Then work on executing the statement.

Test input files

Here are test input files that you should run for each of the statements and the expected runtime output. You may want to create your own simpler tests before trying these. In particular, **`TestIf.txt`** has some complicated expressions that will require modifications to the expression parser and executor. Click on the links to download the files.

You can use the online Pascal compilers to verify the output of the test files. But you'll need to add variable declarations to turn them into valid Pascal programs.

TestWhile.txt

```
program TestWhile;

begin
    i := 1;
    while i <= 5 do begin
        write('i = '); writeln(i);
        i := i + 1
    end;

    writeln;

    i := 1;
    while i <= 5 do begin
        j := 10;

        while j <= 30 do begin
            write('i = '); write(i);
            write(', j = '); writeln(j);
            j := j + 10
        end;

        i := i + 1
    end
```

TestWhile.out.txt

```
i = 1
i = 2
i = 3
i = 4
i = 5

i = 1, j = 10
i = 1, j = 20
i = 1, j = 30
i = 2, j = 10
i = 2, j = 20
i = 2, j = 30
i = 3, j = 10
i = 3, j = 20
i = 3, j = 30
i = 4, j = 10
i = 4, j = 20
i = 4, j = 30
i = 5, j = 10
i = 5, j = 20
i = 5, j = 30
```

TestIf.txt

```
program TestIf;

begin
    i := 1;
    j := 2;

    IF i = j THEN x := 3.14
             ELSE x := -5;

    IF i <> j THEN y := 3.14
              ELSE y := -5;

    write('i = '); write(i:3);
    write(', j = '); write(j:3);
    write(', x = '); write(x:5:2);
    write(', y = '); writeln(y:5:2);

    IF i = j THEN BEGIN
        x := -7
    END
    ELSE BEGIN
        x := 8;
    END;

    IF i <> j THEN BEGIN
        y := 14
    END
    ELSE BEGIN
        y := -2;
    END;

    write('i = ');   write(i:3);
    write(', j = '); write(j:3);
    write(', x = '); write(x:5:2);
    write(', y = '); writeln(y:5:2);

    IF i = j THEN x := 55.55
             ELSE BEGIN
                 x := 77.77;
                 y := 88.88;
             END;

    write('i = ');   write(i:3);
    write(', j = '); write(j:3);
    write(', x = '); write(x:5:2);
    write(', y = '); writeln(y:5:2);
```

continued ...

```
    k := 10;

    if i = j then i := 33
             else if not (i <= j) then i := 44
                                  else if i = j then i := 55
                                                else i := 6;

    write('i = ');   write(i:3);
    write(', j = '); write(j:3);
    write(', x = '); write(x:5:2);
    write(', y = '); writeln(y:5:2);
    write('k = ');   writeln(k:3);

    if not (i <= j) then if i div 22 <= j then j := 9 else j := -9;

    write('i = ');   write(i:3);
    write(', j = '); write(j:3);
    write(', x = '); write(x:5:2);
    write(', y = '); writeln(y:5:2);
    write('k = ');   writeln(k:3);

    if i = j then if i <= j then k := 11 else k := 12;

    write('i = ');   write(i:3);
    write(', j = '); write(j:3);
    write(', x = '); write(x:5:2);
    write(', y = '); writeln(y:5:2);
    write('k = ');   writeln(k:3);

    writeln;
    x := i + j + k - x - y;
    write('x = '); writeln(x:5:2);

    writeln;
    if not (i = j) and (i < j) and (i <> j) and (x < y) then write('Good-bye');
    if not (i < j) or (x <> y) then if i > j/2 then if i <> j then if x < 5*y then write(', world!');

    writeln;
    writeln('Done!');
end.
```

TestIf.out.txt

```
i =   1, j =   2, x = -5.00, y =  3.14
i =   1, j =   2, x =  8.00, y = 14.00
i =   1, j =   2, x = 77.77, y = 88.88
i =   6, j =   2, x = 77.77, y = 88.88
k =  10
i =   6, j =   9, x = 77.77, y = 88.88
k =  10
i =   6, j =   9, x = 77.77, y = 88.88
k =  10

x = -141.65

Good-bye, world!
Done!
```

TestFor.txt

```
program TestFor;

begin
    for i := 1 to 5 do begin
        write('i = '); writeln(i);
    end;
    writeln;

    for i := 5 downto 1 do begin
        write('i = '); writeln(i);
    end;
    writeln;

    for i := 1 to 3 do begin
        for j := 4 downto 1 do begin
            write('i = '); write(i);
            write(', j = '); writeln(j);
        END
    end;

    writeln;
end.
```

TestFor.out.txt

```
i = 1
i = 2
i = 3
i = 4
i = 5

i = 5
i = 4
i = 3
i = 2
i = 1

i = 1, j = 4
i = 1, j = 3
i = 1, j = 2
i = 1, j = 1
i = 2, j = 4
i = 2, j = 3
i = 2, j = 2
i = 2, j = 1
```

TestCase.txt

```
PROGRAM TestCase;

BEGIN
    i := 3; even := -999; odd := -999; prime := -999;

    CASE i+1 OF
        1:       j := i;
        -8:      j := 8*i;
        5, 7, 4: j := 574*i;
    END;

    write('j = '); writeln(j);
    writeln;

    FOR i := -5 TO 15 DO BEGIN
        CASE i OF
            2: BEGIN even := i; prime := i END;
            -4, -2, 0, 4, 6, 8, 10, 12, 14: even := i;
            -5, -3, -1, 1, 3, 5,
            7, 9, 11, 13, 15:  BEGIN
                                   odd := i;
                                   CASE i OF
                                       2, 3, 5, 7, 11, 13: prime := i
                                   END
                               END
        END;

        write('i =');  write(i:3);
        write(', even = ');   IF even  <> -999 THEN write(even:3)  ELSE write('...');
        write(', odd = ');    IF odd   <> -999 THEN write(odd:3)   ELSE write('...');
        write(', prime = ');  IF prime <> -999 THEN write(prime:3) ELSE write('...');
        writeln;

        even := -999; odd := -999; prime := -999
    END;

    writeln;
    writeln('Done!')
END.
```

TestCase.out.txt

```
j = 1722

i = -5, even = ..., odd =  -5, prime = ...
i = -4, even =  -4, odd = ..., prime = ...
i = -3, even = ..., odd =  -3, prime = ...
i = -2, even =  -2, odd = ..., prime = ...
i = -1, even = ..., odd =  -1, prime = ...
i =  0, even =   0, odd = ..., prime = ...
i =  1, even = ..., odd =   1, prime = ...
i =  2, even =   2, odd = ..., prime =   2
i =  3, even = ..., odd =   3, prime =   3
i =  4, even =   4, odd = ..., prime = ...
i =  5, even = ..., odd =   5, prime =   5
i =  6, even =   6, odd = ..., prime = ...
i =  7, even = ..., odd =   7, prime =   7
i =  8, even =   8, odd = ..., prime = ...
i =  9, even = ..., odd =   9, prime = ...
i = 10, even =  10, odd = ..., prime = ...
i = 11, even = ..., odd =  11, prime =  11
i = 12, even =  12, odd = ..., prime = ...
i = 13, even = ..., odd =  13, prime =  13
i = 14, even =  14, odd = ..., prime = ...
i = 15, even = ..., odd =  15, prime = ...

Done!
```

What to submit to Canvas

A zip file that contains:

- All of your Java source files and any extra input test programs you wrote.
- Cut-and-paste text files of the parse trees of each of the four input test programs that were generated from the **`-parse`** command option.
- Cut-and-paste text files of the runtime output of the above simple Pascal test programs that were generated from the **`-execute`** command option.

Submit to **Assignment #3: Simple Pascal Interpreter**

There should be only one submission per team.

Rubric

Your submission will be graded according to these criteria:

Criteria	Max points
`WHILE` statement • Parse tree • Runtime output	**25** • 10 • 15
`IF` statement • Parse tree • Runtime output	**25** • 10 • 15
`FOR` statement • Parse tree • Runtime output	**25** • 10 • 15
`CASE` statement • Parse tree • Runtime output	**25** • 10 • 15