A Simple Compiler for Pascal

Rohan Mohapatra 01FB16ECS307
Rohan Talesara 01FB16ECS309
Saahitya Edamadaka 01FB16ECS322

January 27, 2019

Abstract

We are writing a simple compiler for PASCAL Language. We start off by defining the rules and grammar for this particular language. We will continue this project by adding rest of the stages.

1 Lexical Analysis

This is the initial stage of defining and building our compiler. We start off by writing a Context Free Grammar to define the Rules of our Compiler.

1.1 Context Free Grammar

The following structure shows the basic syntax for a **pascal** program:

program { name of the program }
uses {comma delimited names of libraries you use}
const {global constant declaration block}
var {global variable declaration block}
begin { main program block starts}
...
end. { the end of main program block }

We start by defining the **Context Free Grammar** for Pascal Language:

$$<program> \rightarrow <program\_heading> <block> .$$

$$<program\_heading> \rightarrow program <identifier>$$

$$\begin{aligned} <block> \rightarrow\ & <uses\_block><constant\_block> \\ & <type\_block><variable\_block> \\ & <execution\_block> \end{aligned}$$

```
<uses_block> → uses <identifier> ; | λ

<constant_block> → const<newline> <const_definition>
<const_defnition> → <identifier> = <constant>
<constant> →    integer |
                boolean |
                string


<type_block> → type <newline> <type_definition> | λ
<type_definition> → <identifier>  = <type>

<variable_block> → λ | var <new line> <decl_stmts>
<decl_stmts> → <decl_stmt> <new line> <decl_stmts> | <decl_stmt>
<decl_stmt> → <identifier> : <type> ;


<execution_block> → begin <newline> <exec_body> <newline> end .
<exec_body> → <assignment_stmts> <newline> <exec_body>
                | <if_statement> <newline> <exec_body>
                | <while_statement> <newline> <exec_body>
                | λ

<assignment_stmts> → <assignment_stmts> <newline>
                       <assignment_statement>
                     | <assignment_stmt>

<assigment_stmt> → <identifier> : = <value>


<if_statement> → if <cond_stmt> then <newline> <exec_body>
                   | if <cond_stmt> then <newline> <exec_body>
                     else <newline> <exec_body>



<while_statement> → while <cond_stmt> do <newline> <exec_body>


<cond_stmt> → <expression>
                | <expression> <operator> <expression>
<operator> → && | || | < | <= | >= | > | = | == | + | - | * | /
              <expression> → <identifier> | <value>
                | ( <expression> )
                | <expression> <operator> <expression>
```

$$<type> \rightarrow character \mid integer \mid real \mid boolean \mid string$$