

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:

$$\langle program \rangle \rightarrow \langle program_heading \rangle \langle block \rangle .$$
$$\langle program_heading \rangle \rightarrow program \langle identifier \rangle$$
$$\begin{aligned} \langle block \rangle \rightarrow & \langle uses_block \rangle \langle constant_block \rangle \\ & \langle type_block \rangle \langle variable_block \rangle \\ & \langle execution_block \rangle \end{aligned}$$

$\langle \text{uses_block} \rangle \rightarrow \text{uses } \langle \text{identifier} \rangle ; \mid \lambda$

$\langle \text{constant_block} \rangle \rightarrow \text{const} \langle \text{newline} \rangle \langle \text{const_definition} \rangle$

$\langle \text{const_definition} \rangle \rightarrow \langle \text{identifier} \rangle = \langle \text{constant} \rangle$

$\langle \text{constant} \rangle \rightarrow$
 $\text{integer} \mid$
 $\text{boolean} \mid$
 string

$\langle \text{type_block} \rangle \rightarrow \text{type } \langle \text{newline} \rangle \langle \text{type_definition} \rangle \mid \lambda$

$\langle \text{type_definition} \rangle \rightarrow \langle \text{identifier} \rangle = \langle \text{type} \rangle$

$\langle \text{variable_block} \rangle \rightarrow \lambda \mid \text{var } \langle \text{new line} \rangle \langle \text{decl_stmts} \rangle$

$\langle \text{decl_stmts} \rangle \rightarrow \langle \text{decl_stmt} \rangle \langle \text{new line} \rangle \langle \text{decl_stmts} \rangle \mid \langle \text{decl_stmt} \rangle$

$\langle \text{decl_stmt} \rangle \rightarrow \langle \text{identifier} \rangle : \langle \text{type} \rangle ;$

$\langle \text{execution_block} \rangle \rightarrow \text{begin } \langle \text{newline} \rangle \langle \text{exec_body} \rangle \langle \text{newline} \rangle \text{end } .$

$\langle \text{exec_body} \rangle \rightarrow$
 $\langle \text{assignment_stmts} \rangle \langle \text{newline} \rangle \langle \text{exec_body} \rangle$
 $\mid \langle \text{if_statement} \rangle \langle \text{newline} \rangle \langle \text{exec_body} \rangle$
 $\mid \langle \text{while_statement} \rangle \langle \text{newline} \rangle \langle \text{exec_body} \rangle$
 $\mid \lambda$

$\langle \text{assignment_stmts} \rangle \rightarrow \langle \text{assignment_stmts} \rangle \langle \text{newline} \rangle$
 $\langle \text{assignment_statement} \rangle$
 $\mid \langle \text{assignment_stmt} \rangle$

$\langle \text{assignment_stmt} \rangle \rightarrow \langle \text{identifier} \rangle : = \langle \text{value} \rangle$

$\langle \text{if_statement} \rangle \rightarrow$
 $\text{if } \langle \text{cond_stmt} \rangle \text{ then } \langle \text{newline} \rangle \langle \text{exec_body} \rangle$
 $\mid \text{if } \langle \text{cond_stmt} \rangle \text{ then } \langle \text{newline} \rangle \langle \text{exec_body} \rangle$
 $\text{else } \langle \text{newline} \rangle \langle \text{exec_body} \rangle$

$\langle \text{while_statement} \rangle \rightarrow \text{while } \langle \text{cond_stmt} \rangle \text{ do } \langle \text{newline} \rangle \langle \text{exec_body} \rangle$

$\langle \text{cond_stmt} \rangle \rightarrow \langle \text{expression} \rangle$

$\langle \text{expression} \rangle \rightarrow$
 $\langle \text{expression} \rangle \langle \text{operator} \rangle \langle \text{expression} \rangle$
 $\langle \text{operator} \rangle \rightarrow \mathcal{E}\mathcal{E} \mid \mid \mid < \mid \leq \mid \geq \mid > \mid = \mid == \mid + \mid - \mid * \mid /$
 $\langle \text{expression} \rangle \rightarrow \langle \text{identifier} \rangle \mid \langle \text{value} \rangle$
 $\mid (\langle \text{expression} \rangle)$
 $\mid \langle \text{expression} \rangle \langle \text{operator} \rangle \langle \text{expression} \rangle$

$\langle type \rangle \rightarrow character \mid integer \mid real \mid boolean \mid string$