# 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:

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

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

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

$$<uses\_block> \rightarrow uses <identifier> ; \mid \lambda$$

$$<constant\_block> \rightarrow const<newline> <const\_definition>$$
$$<const\_defnition> \rightarrow <identifier> = <constant>$$
$$\begin{aligned} <constant> \rightarrow \quad & integer \\ & \mid boolean \\ & \mid string \end{aligned}$$

$$<type\_block> \rightarrow type <newline> <type\_definition> \mid \lambda$$
$$<type\_definition> \rightarrow <identifier> = <type>$$

$$<variable\_block> \rightarrow \lambda \mid var <new\ line> <decl\_stmts>$$
$$<decl\_stmts> \rightarrow <decl\_stmt> <new\ line> <decl\_stmts> \mid <decl\_stmt>$$
$$<decl\_stmt> \rightarrow <identifier> : <type> ;$$

$$<execution\_block> \rightarrow begin <newline> <exec\_body> <newline> end .$$
$$\begin{aligned} <exec\_body> \rightarrow & <assignment\_stmts> <newline> <exec\_body> \\ & \mid <if\_statement> <newline> <exec\_body> \\ & \mid <while\_statement> <newline> <exec\_body> \\ & \mid \lambda \end{aligned}$$

$$\begin{aligned} <assignment\_stmts> \rightarrow & <assignment\_stmts> <newline> \\ & <assignment\_statement> \\ & \mid <assignment\_stmt> \end{aligned}$$

$$<assigment\_stmt> \rightarrow <identifier> : = <value>$$

$$\begin{aligned} <if\_statement> \rightarrow & if <cond\_stmt> then <newline> <exec\_body> \\ & \mid if <cond\_stmt> then <newline> <exec\_body> \\ & \quad else <newline> <exec\_body> \end{aligned}$$

$$<while\_statement> \rightarrow while <cond\_stmt> do <newline> <exec\_body>$$

$$\begin{aligned} <cond\_stmt> \rightarrow & <expression> \\ & \mid <expression> <operator> <expression> \end{aligned}$$
$$<operator> \rightarrow \&\& \mid || \mid < \mid <= \mid >= \mid > \mid = \mid == \mid + \mid - \mid * \mid /$$
$$\begin{aligned} & <expression> \rightarrow <identifier> \mid <value> \\ & \mid ( <expression> ) \\ & \mid <expression> <operator> <expression> \end{aligned}$$

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