Pascal Grammar

Jawad Ahmed

20th January 2024

This document details a hierarchical EBNF grammar for Pascal per the ISO 7185:1990 standard.

0 Programs

⟨program⟩ ::= ⟨program heading⟩ ‘`;`’ ⟨block⟩ ‘`.`’

⟨program heading⟩ ::= ‘`program`’ ⟨identifier⟩ [⟨program param list⟩]

⟨program param list⟩ ::= ‘`(`’ ⟨identifier list⟩ ‘`)`’

⟨block⟩ ::= [⟨label decl part⟩] [⟨const def part⟩] [⟨type def part⟩]
[⟨var decl part⟩] ⟨routine part⟩ ⟨stmt part⟩

⟨label decl part⟩ ::= ‘`label`’ ⟨digit seq⟩ { ‘`,`’ ⟨digit seq⟩ } ‘`;`’

⟨const def part⟩ ::= ‘`const`’ ⟨const def⟩ ‘`;`’ { ⟨const def⟩ ‘`;`’ }

⟨type def part⟩ ::= ‘`type`’ ⟨type def⟩ ‘`;`’ { ⟨type def⟩ ‘`;`’ }

⟨var decl part⟩ ::= ‘`var`’ ⟨var decl⟩ ‘`;`’ { ⟨var decl⟩ ‘`;`’ }

⟨routine decl part⟩ ::= { (⟨proc decl⟩ | ⟨func decl⟩) ‘`;`’ }

⟨stmt part⟩ ::= ⟨compound stmt⟩

1 Declarations

⟨const def⟩ ::= ⟨ident⟩ ‘`=`’ ⟨const⟩

⟨type def⟩ ::= ⟨ident⟩ ‘`=`’ ⟨type⟩

⟨var decl⟩ ::= ⟨ident list⟩ ‘`:`’ ⟨type⟩

⟨proc decl⟩ ::= ⟨proc heading⟩ ‘`;`’ ⟨block⟩
| ⟨proc heading⟩ ‘`;`’ ⟨directive⟩
| ⟨proc identification⟩ ‘`;`’ ⟨block⟩

⟨func decl⟩ ::= ⟨func heading⟩ ‘`;`’ ⟨block⟩
| ⟨func heading⟩ ‘`;`’ ⟨directive⟩
| ⟨func identification⟩ ‘`;`’ ⟨block⟩

⟨proc heading⟩ ::= ‘`procedure`’ ⟨ident⟩ [⟨formal param list⟩]

⟨proc identification⟩ ::= ‘`procedure`’ ⟨proc ident⟩

⟨func heading⟩ ::= ‘`function`’ ⟨ident⟩ [⟨formal param list⟩] ‘`:`’ ⟨result type⟩

⟨func identification⟩ ::= 'function' ⟨func ident⟩

⟨formal param list⟩ ::= '(' ⟨formal param section⟩ { ';' ⟨formal param section⟩ } ')'

⟨formal param section⟩ ::= ⟨value param spec⟩
| ⟨var param spec⟩
| ⟨proc param spec⟩
| ⟨func param spec⟩

2 Parameters

⟨value param spec⟩ ::= ⟨ident list⟩ ':'
(⟨type ident⟩ | ⟨conformant array schema⟩)

⟨var param spec⟩ ::= 'var' ⟨ident list⟩ ':'
(⟨type ident⟩ | ⟨conformant array schema⟩)

⟨proc param spec⟩ ::= ⟨proc heading⟩

⟨func param spec⟩ ::= ⟨func heading⟩

⟨conformant array schema⟩ ::= ⟨packed conformant array schema⟩
| ⟨unpacked conformant array schema⟩

⟨packed conformant array schema⟩ ::= 'packed' 'array' '[' ⟨index type spec⟩ ']' 'of' ⟨type ident⟩

⟨unpacked conformant array schema⟩ ::= 'array' '[' ⟨index type spec⟩ { ';' ⟨index type spec⟩ } ']'
'of' (⟨type ident⟩ | ⟨conformant array schema⟩)

⟨index type spec⟩ ::= ⟨ident⟩ '..' ⟨ident⟩ ':' ⟨ordinal type ident⟩

3 Statements

⟨compound stmt⟩ ::= 'begin' ⟨stmt seq⟩ 'end'

⟨stmt seq⟩ ::= ⟨stmt⟩ { ';' ⟨stmt⟩ }

⟨stmt⟩ ::= [⟨label⟩ ':'] (⟨simple stmt⟩ | ⟨structured stmt⟩)

⟨simple stmt⟩ ::= ⟨empty stmt⟩
| ⟨assign stmt⟩
| ⟨proc stmt⟩
| ⟨goto stmt⟩

⟨structured stmt⟩ ::= ⟨compound stmt⟩
| ⟨conditional stmt⟩
| ⟨repetitive stmt⟩
| ⟨with stmt⟩

⟨conditional stmt⟩ ::= ⟨if stmt⟩
| ⟨case stmt⟩

⟨repetitive stmt⟩ ::= ⟨while stmt⟩
| ⟨repeat stmt⟩
| ⟨for stmt⟩

⟨empty stmt⟩ ::=

⟨assign stmt⟩ ::= (⟨var⟩ | ⟨func ident⟩) ‘:=’ ⟨expr⟩

⟨procedure stmt⟩ ::= ⟨proc ident⟩ [⟨actual param list⟩ | ⟨write param list⟩]

⟨goto stmt⟩ ::= ‘goto’ ⟨label⟩

⟨if stmt⟩ ::= ‘if’ ⟨bool expr⟩ ‘then’ ⟨stmt⟩ [‘else’ ⟨stmt⟩]

⟨case stmt⟩ ::= ‘case’ ⟨case index⟩ ‘of’ ⟨case⟩ { ‘;’ ⟨case⟩ } [‘;’] ‘end’

⟨repeat stmt⟩ ::= ‘repeat’ ⟨stmt seq⟩ ‘until’ ⟨bool expr⟩

⟨while stmt⟩ ::= ‘while’ ⟨bool expr⟩ ‘do’ ⟨stmt⟩

⟨for stmt⟩ ::= ‘for’ ⟨control var⟩ ‘:=’ ⟨ordinal expr⟩ (‘to’ | ‘downto’)
⟨ordinal expr⟩ ‘do’ ⟨stmt⟩

⟨with stmt⟩ ::= ‘with’ ⟨record var list⟩ ‘do’ ⟨stmt⟩

⟨record var list⟩ ::= ⟨record var⟩ { ‘,’ ⟨record var⟩ }

⟨case index⟩ ::= ⟨ordinal expr⟩

⟨case⟩ ::= ⟨const⟩ { ‘,’ ⟨const⟩ } ‘:’ ⟨stmt⟩

⟨control var⟩ ::= ⟨var ident⟩

4 Types

⟨type⟩ ::= ⟨simple type⟩
| ⟨structured type⟩
| ⟨pointer type⟩

⟨simple type⟩ ::= ⟨ordinal type⟩
| ⟨real type ident⟩

⟨structured type⟩ ::= [‘packed’] ⟨unpacked structured type⟩
| ⟨structured type ident⟩

⟨pointer type⟩ ::= (‘^’ | ‘↑’) ⟨domain type⟩
| ⟨pointer type ident⟩

⟨ordinal type⟩ ::= ⟨enumerated type⟩
| ⟨subrange type⟩
| ⟨ordinal type ident⟩

⟨unpacked structured type⟩ ::= ⟨array type⟩
| ⟨record type⟩
| ⟨set type⟩
| ⟨file type⟩

⟨domain type⟩ ::= ⟨type ident⟩

⟨enumerated type⟩ ::= ‘(’ ⟨ident list⟩ ‘)’

⟨subrange type⟩ ::= ⟨const⟩ ‘..’ ⟨const⟩

⟨array type⟩ ::= ‘array’ ‘[’ ⟨index type⟩ { ‘,’ ⟨index type⟩ } ‘]’ ‘of’ ⟨component type⟩

⟨record type⟩ ::= 'record' ⟨field list⟩ 'end'

⟨set type⟩ ::= 'set' 'of' ⟨base type⟩

⟨file type⟩ ::= 'file' 'of' ⟨component type⟩

⟨index type⟩ ::= ⟨ordinal type⟩

⟨component type⟩ ::= ⟨type⟩

⟨base type⟩ ::= ⟨ordinal type⟩

⟨result type⟩ ::= ⟨ordinal type ident⟩
| ⟨real type ident⟩
| ⟨pointer type ident⟩

⟨field list⟩ ::= [(⟨fixed part⟩ [';' ⟨variant part⟩] | ⟨variant part⟩) [';']]

⟨fixed part⟩ ::= ⟨record section⟩ { ';' ⟨record section⟩ }

⟨variant part⟩ ::= 'case' ⟨variant selector⟩ 'of' ⟨variant⟩ { ';' ⟨variant⟩ }

⟨record section⟩ ::= ⟨ident list⟩ ':' ⟨type⟩

⟨variant selector⟩ ::= [⟨tag field⟩ ':'] ⟨tag type⟩

⟨variant⟩ ::= ⟨const⟩ { ',' ⟨const⟩ } ':' '(' ⟨field list⟩ ')'

⟨tag type⟩ ::= ⟨ordinal type ident⟩

⟨tag field⟩ ::= ⟨ident⟩

⟨const⟩ ::= [⟨sign⟩] (⟨unsigned const⟩ | ⟨const ident⟩)

5 Expressions

⟨expr⟩ ::= ⟨simple expr⟩ [⟨relational op⟩ ⟨simple expression⟩]

⟨simple expr⟩ ::= [⟨sign⟩] ⟨term⟩ { ⟨adding op⟩ ⟨term⟩ }

⟨term⟩ ::= ⟨factor⟩ { ⟨multiplying op⟩ ⟨factor⟩ }

⟨factor⟩ ::= ⟨unsigned const⟩
| ⟨bound ident⟩
| ⟨var⟩
| ⟨set constructor⟩
| ⟨function designator⟩
| 'not' ⟨factor⟩
| '(' ⟨expr⟩ ')'

⟨relational op⟩ ::= '=' | '<>' | '<' | '>' | '≤' | '≥' | 'in'

⟨adding op⟩ ::= '+' | '-' | 'or'

⟨multiplying op⟩ ::= '*' | '/' | 'div' | 'mod' | 'and'

⟨unsigned const⟩ ::= ⟨unsigned number⟩
| ⟨char string⟩
| ⟨const ident⟩
| 'nil'

⟨function designator⟩ ::= ⟨function ident⟩ [⟨actual param list⟩]

⟨var⟩ ::= ⟨entire var⟩
| ⟨component var⟩
| ⟨identified var⟩
| ⟨buffer var⟩

⟨entire var⟩ ::= ⟨var ident⟩

⟨component var⟩ ::= ⟨indexed var⟩
| ⟨field designator⟩

⟨identified var⟩ ::= ⟨pointer var⟩ ('^' | '↑')

⟨buffer var⟩ ::= ⟨file var⟩ ('^' | '↑')

⟨indexed var⟩ ::= ⟨array var⟩ '[' ⟨index⟩ { ',' ⟨index⟩ } ']'

⟨field designator⟩ ::= [⟨record var⟩ '.'] ⟨field ident⟩

⟨set constructor⟩ ::= '[' [⟨element description⟩ { ',' ⟨element description⟩ }] ']'

⟨element description⟩ ::= ⟨ordinal expr⟩ ['..' ⟨ordinal expr⟩]

⟨actual param list⟩ ::= '(' ⟨actual param⟩ { ',' ⟨actual param⟩ } ')'

⟨actual param⟩ ::= ⟨expr⟩
| ⟨var⟩
| ⟨proc ident⟩
| ⟨func ident⟩

⟨write param list⟩ ::= '(' (⟨file var⟩ | ⟨write param⟩) { ',' ⟨write param⟩ } ')'

⟨write param⟩ ::= ⟨expr⟩ [':' ⟨integer expr⟩ [':' ⟨integer expr⟩]]

6 Tokens

⟨array var⟩ ::= ⟨var⟩

⟨record var⟩ ::= ⟨var⟩

⟨file var⟩ ::= ⟨var⟩

⟨pointer var⟩ ::= ⟨var⟩

⟨integer expr⟩ ::= ⟨ordinal expr⟩

⟨bool expr⟩ ::= ⟨ordinal expr⟩

⟨index⟩ ::= ⟨ordinal expr⟩

⟨ordinal expr⟩ ::= ⟨expr⟩

⟨pointer type ident⟩ ::= ⟨type ident⟩

⟨structured type ident⟩ ::= ⟨type ident⟩

⟨ordinal type ident⟩ ::= ⟨type ident⟩

⟨real type ident⟩ ::= ⟨type ident⟩

⟨const ident⟩ ::= ⟨ident⟩

⟨type ident⟩ ::= ⟨ident⟩

⟨var ident⟩ ::= ⟨ident⟩

⟨field ident⟩ ::= ⟨ident⟩

⟨proc ident⟩ ::= ⟨ident⟩

⟨func ident⟩ ::= ⟨ident⟩

⟨bound ident⟩ ::= ⟨ident⟩

⟨unsigned number⟩ ::= ⟨unsigned integer⟩
| ⟨unsigned real⟩

⟨ident list⟩ ::= ⟨ident⟩ { ‘,’ ⟨ident⟩ }

⟨ident⟩ ::= ⟨letter⟩ { ⟨letter⟩ | ⟨digit⟩ }

⟨directive⟩ ::= ⟨letter⟩ { ⟨letter⟩ | ⟨digit⟩ }

⟨label⟩ ::= ⟨digit seq⟩

⟨unsigned integer⟩ ::= ⟨digit seq⟩

⟨unsigned real⟩ ::= ⟨digit seq⟩ ‘.’ ⟨digit seq⟩ [‘e’ ⟨scale factor⟩]
| ⟨digit seq⟩ ‘e’ ⟨scale factor⟩

⟨scale factor⟩ ::= [⟨sign⟩] ⟨digit seq⟩

⟨sign⟩ ::= ‘+’ | ‘−’

⟨char string⟩ ::= ‘'’ ⟨string element⟩ { ⟨string element⟩ } ‘'’

⟨digit seq⟩ ::= ⟨digit⟩ { ⟨digit⟩ }

⟨letter⟩ ::= ‘a’ | ‘b’ | ‘c’ | ‘d’ | ‘e’ | ‘f’ | ‘g’ | ‘h’ | ‘i’ | ‘j’ | ‘k’ | ‘l’ | ‘m’ | ‘n’ | ‘o’ | ‘p’ | ‘q’ | ‘r’ |
‘s’ | ‘t’ | ‘u’ | ‘v’ | ‘w’ | ‘x’ | ‘y’ | ‘z’

⟨digit⟩ ::= ‘0’ | ‘1’ | ‘2’ | ‘3’ | ‘4’ | ‘5’ | ‘6’ | ‘7’ | ‘8’ | ‘9’

⟨string element⟩ ::= ‘''’ | ⟨any char except ‘'’⟩

7 Key

Abbreviation	Expansion
decl	declaration
def	definition
const	constant
var	variable
routine	procedure and function
stmt	statement
seq	sequence
ident	identifier
proc	procedure
func	function
param	parameter
spec	specification
assign	assignment
bool	Boolean
op	operator
char	character