

# Grammar of The $\mu$ Pascal Programming Language

## Notation:

<b>()</b>	grouping
<b> </b>	alternatives ("or")
<b>[]</b>	0 or 1 occurrences
<b>{ }</b>	0 or more occurrences
<i>italic</i>	nonterminals ( <i>pojmovi</i> )
<b>bold</b>	terminals ( <i>simboli</i> )

## Patterns

digit

→ "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9"

letter

→ "\_" | "a" | "A" | "b" | "B" | "c" | "C" | "d" | "D" | "e" | "E" | "f"  
| "F" | "g" | "G" | "h" | "H" | "i" | "I" | "j" | "J" | "k" | "K" | "l"  
| "L" | "m" | "M" | "n" | "N" | "o" | "O" | "p" | "P" | "q" | "Q" | "r"  
| "R" | "s" | "S" | "t" | "T" | "u" | "U" | "v" | "V" | "w" | "W" | "x"  
| "X" | "y" | "Y" | "z" | "Z"

constant

→ digit +

identifier

→ letter ( letter | digit ) \*

## Productions

*program*

→ **"program"** identifier **;"** [ **"var"** variable\_list ] [ procedure\_list ]  
**"begin"** statement\_list **"end"**

*variable\_list*

→ variable { **;"** variable }

*variable*

→ identifier **:"** **"int"**

*procedure\_list*

→ procedure { **;"** procedure }

*procedure*

→ **"procedure"** identifier [ **"("** variable\_list **)"** ] [ **"var"** variable\_list ]  
**"begin"** statement\_list **"end"**

*statement\_list*

→ [ statement { **;"** statement } ]

*statement*

→ assignment\_or\_procedure\_call\_statement  
| if\_statement  
| while\_statement  
| **"begin"** statement\_list **"end"**

*assignment\_or\_procedure\_call\_statement*

→ identifier ( **:"=** expression | [ **"("** expression { **","** expression } **)"** ] )

*expression*

→ [  **"+" | "-"**  ] operand { (  **"\*" | "/"**  ) operand }  
{ (  **"+" | "-"**  ) operand { (  **"\*" | "/"**  ) operand } }

*operand*

→ constant  
| identifier  
| "(" expression ")"

*if\_statement*

→ "if" condition "then" statement [ "else" statement ]

*condition*

→ expression ( "<" | ">" | "<=" | ">=" | "=" | "<>" ) expression

*while\_statement*

→ "while" condition "do" statement

Comment: { a comment }

Space and newline are symbol separators.

Symbol	Token
eof	EOF
" , "	COMMA
" ; "	SEMICOLON
" ( "	LEFT PAR
" ) "	RIGTH PAR
" + "	PLUS
" - "	MINUS
" * "	MULTIPLY
" / "	DIVIDE
" := "	ASSIGN
" : "	COLON
" = "	EQ
" < = "	LE
" < > "	NE
" < "	LT
" > = "	GE
" > "	GT
letter ( letter   digit ) *	IDENTIFIER
"program"	PROGRAM
"var"	VAR
"proc"	PROC
"begin"	BEGIN
"end"	END
"if"	IF
"then"	THEN
"else"	ELSE
"while"	WHILE
"do"	DO
"int"	INT
digit +	CONSTANT

### Operator Precedence

All operators on the same line have the same precedence. The first line has the highest precedence.

()	the highest precedence
+ -	unary
* /	
+ -	arithmetic
< <= >= > = < >	
:=	the lowest precedence