DISEÑO DE COMPILADORES
BNF EC-Pascal

CAMPUS QUERÉTARO

---

1. Programs and blocks

programa ::= <program-heading> “**;**” <program-block> “**.**”

program-heading ::= **program** <identifier> [“**(**” <program-parameters> “**)**”]

programa-parameters ::= <identifier-list>

program-block ::= <block>

body ::= [<constant-declaration-part>]
[<variable-declaration-part>]
[<procedure-and-function-declaration-part>]
<statement-part>

constant-declaration-part ::= **constant** <constant-definition> “**;**”
{ <constant-definition> “**;**” }

constant-definition ::= <identifier> “**=**” <constant>

variable-declaration-part := **var** <variable-declaration> “**;**” { <variable-declaration> “**;**” }

variable-declaration ::= <identifier-list> “**:**” <type>

procedure-and-function-part ::=
{ (<procedure-declaration> | <procedure-declaration>) “**;**” }

procedure-declaration ::= <procedure-heading> “**;**” <procedure-body>

procedure-body ::= <block>

function-declaration ::= <function-heading> “**;**” <function-body>

function-body ::= <block>

statement-part ::= **begin** <statement-sequence> **end** “**.**”

2. Procedure and Functions Definitios

procedure-heading ::= **procedure-identifier** <identifier> [<formal-parameter-list>]

procedure-identifier ::= <identifier>

function-heading ::= **function** <function-identifier> [<formal-parameter-list>] “**:**”
<result-type>

function-identifier ::= <identifier>

result-type ::= <type>

formal-parameter-list ::=
"**(**" <formal-parameter-section> { "**;**" <formal-parameter-section> } "**)**"

formal-parameter-section ::=
<value-parameter-section> | <variable-parameter-section>

value-parameter-section ::= <identifier-list> "**:**" <parameter-type>

variable-parameter-section ::= **var** <identifier-list> "**:**" <parameter-type>

parameter-type ::= <type>

3. Statements

statement-sequence ::= <statement> { "**;**" <statement> }

statement ::= <simple-statement> | <structured-statement>

simple-statement ::= [<assignment-statement> | <procedure-statement>]

assigment-statement ::=
(<variable-identifier> | <function-identifier>) "**:=**" <expression>

procedure-statement ::= <procedure-identifier> [<actual-parameter-list>]

structured-statement ::= <compound-statement> | <repetitive-statement> |
<conditional-statement>

compound-statement ::= **begin** <statement-sequence> **end** "**.**"

repetitive-statement ::= <while-statement> | <repeat-statement> | <for-statement>

while-statement ::= **while** <expression> **do** <statement>

repeat-statement ::= **repeat** <statement-sequence> **until** <expression>

for-statement ::=
for <variable-identifier> "**:=**" <initial-expression> (**to**|**downto**)
<final-expression> **do** <statement>

initial-expression ::= <expression>

final-expression ::= <expression>

conditional-statement ::= <if-statement>

if-statement ::= **if** <expression> **then** <statement> [**else** <statement>]

actual-parameter-list ::= "**(**" actual-parameter { "**,**" <actual-parameter> } "**)**"

actual-parameter ::= <actual-value> | <actual-variable> | <actual-procedure> |
<actual-function>

actual-value ::= <expression>

actual-variable ::= <variable-identifier>

actual-procedure ::= <procedure-identifier>

actual-function ::= <function-identifier>

writeln-statement ::= “**writeln**” “**(**” [<element-list>] “**)**”

element-list ::= <element> { “**,**” <element> }

element ::= <variable> | <number> | <string> | <constant>

readln-statement ::= “**readln**” [“**(**” <identifier-list> “**)**”]

4. Expressions

expression ::= <simple-expression> [<relational-operator> <simple-expression>]

simple-expression ::= [<sign>] <term> { <addition-operator> <term> }

term ::= <factor> { <multiplication-operator> <factor> }

factor ::= <variable> | <number> | <string> | <constant-identifier> |
“**(**” <expression> “**)**” | “**not**” <factor>

relational-operator ::= “**=**” | “**<>**” | “**<**” | “**<=**” | “**>**” | “**>=**”

addition-operator ::= “**+**” | “**-**” | “**or**”

multiplication-operator ::= “*****” | “/” | “**div**” | “**mod**” | “**and**”

variable ::= <variable-identifier>

5. Variable and Identifier Categories

actual-variable ::= <variable>

constant-identifier ::= <identifier>

variable-identifier ::= <identifier>

procedure-identifier ::= <identifier>

function-identifier ::= <identifier>

type ::= **integer** | **real** | **boolean** | **string**

identifier ::= <letter> { <letter> | <digit> }

6. Low Level Definitions

variable-list ::= <variable> { “**,**” <variable> }

identifier-list ::= <identifier> { “**,**” <identifier> }

number ::= <integer-number> | <real-number>

integer-number ::= <digit-sequence>

real-number ::= <digit-sequence> “**.**” [<unsigned-digit-sequence>] [<scale-factor>] |
<digit-sequence> <scale-factor>

scale-factor ::= (“**E**” | “**e**”) <digit-sequence>

unsigned-digit-sequence ::= <digit> { <digit> }

digit-sequence ::= [<sign>] <unsigned-digit-sequence>

sign ::= “**+**” | “**-**”

letter ::= [“**A**” - “**Z**”] | [“**a**” - “**z**”]

digit ::= [“**0**” - “**9**”]

string ::= “’” <string-character> { string-character } “’”

string-character ::= <any-character-except-quote> | “”

constant ::= [<sign>] (<constant-identifier> | <number>) | <string>