MINI-PASCAL GRAMMAR REFERENCE

MINI-PASCAL COMPILER

Version 0.6

William Mork

Augsburg University

Production Rules

Below are the production rules for each non-terminal symbol in Mini-Pascal grammar as they are employed within compiler project. Each production rule has been given a unique label for ease-of-reference when browsing this project's source files. Refer to page 5 for a basic list of lexical conventions.

---

program
- → **program id ;** declarations subprogram_declarations compound_statement **.**   (a.a)

Identifier_list
- → **id**   (b.a)
- → **id ,** identifier_list   (b.b)

declarations
- → **var** identifier_list **:** type **;** declarations   (c.a)
- → λ   (c.b)

type
- → standard_type   (d.a)
- → **array [num : num] of** standard_type   (d.b)

standard_type
- → **integer**   (e.a)
- → **real**   (e.b)

subprogram_declarations
- → subprogram_declaration **;** subprogram_declarations   (f.a)
- → λ   (f.b)

subprogram_declaration
- → subprogram_head declarations compound_statement   (g.a)

subprogram_head
- → **function id** arguments **:** standard_type **;**   (h.a)
- → **procedure id** arguments **;**   (h.b)

arguments
- → **(** parameter_list **)**   (i.a)
- → λ   (i.b)

parameter_list
- → identifier_list **:** type   (j.a)
- → identifier_list **:** type **;** parameter_list   (j.b)

compound_statement
- → **begin** optional_statements **end**   (k.a)

optional_statements
  → statement_list   (l.a)
  → λ   (l.b)

statement_list
  → statement   (m.a)
  → statement ; statement_list   (m.b)

statement
  → variable **assignop** expression   (n.a)
  → procedure_statement   (n.b)
  → compound_statement   (n.c)
  → **if** expression **then** statement **else** statement (n.d)
  → **while** expression **do** statement   (n.e)
  → **read (id)**   (n.f)
  → **write (** expression **)**   (n.g)
  → **return** expression   (n.h)

variable
  → **id**   (o.a)
  → **id [** expression **]**   (o.b)

procedure_statement
  → **id**   (p.a)
  → **id (** expression_list **)**   (p.b)

expression_list
  → expression   (q.a)
  → expression **,** expression_list   (q.b)

expression
  → simple_expression   (r.a)
  → simple_expression **relop** simple_expression (r.b)

simple_expression
  → term simple_part   (s.a)
  → sign term simple_part   (s.b)

simple_part
  → **addop** term simple_part   (t.a)
  → λ   (t.b)

term
  → factor term_part   (u.a)

term_part
  → **mulop** factor term_part   (v.a)
  → λ   (v.b)

sign

      → **+**   (w.a)

      → **-**   (w.b)

factor

      → **id**   (x.a)

      → **id [** expression **]**   (x.b)

      → **id (** expression_list **)**   (x.c)

      → **num**   (x.d)

      → **(** expression **)**   (x.e)

      → **not** factor   (x.f)

Lexical Conventions

---

1. Comments are surrounded by **{** and **}**, and may not contain a **{**. Comments may appear after any token.

2. Blanks between tokens are optional.

3. Token **id** for identifiers matches a letter followed by letter or digits:
        letter → [a-zA-Z]
        digit → [0-9]
        id → letter (letter | digit)*

4. The ***** indicates that the option in parentheses may be chosen as many times as you wish.

5. Token **num** matches numbers as follows:
        digits → digit digit*
        optional_fraction → . digits | λ
        optional_exponent → (E (+ | - | λ) digits) | λ
        num → digits optional_fraction optional_exponent

6. Keywords are reserved.

7. The relational operators (**relop** characters) are: **=   <>   <   <=   >=   >**

8. The additional operators (**addop** characters) are: **+   -   OR**

9. The multiplicative operators (**mulop** characters) are: ***   /   DIV   MOD   AND**

10. The assignment operator (**assignop**) has the lexeme: **:=**