

# MINI-PASCAL GRAMMAR REFERENCE

---

**MINI-PASCAL COMPILER**

**Version 0.4.1**

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)

→  $\lambda$  (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)

→  $\lambda$  (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)

→  $\lambda$  (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)
- $\lambda$  (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)
- $\lambda$  (t.b)

**term**

- factor term\_part (u.a)

**term\_part**

- **mulop** factor term\_part (v.a)
- $\lambda$  (v.b)

**sign**

- +
- -

(w.a)  
(w.b)

**factor**

- **id**
- **id** [ expression ]
- **id** ( expression\_list )
- **num**
- ( expression )
- **not** factor

(x.a)  
(x.b)  
(x.c)  
(x.d)  
(x.e)  
(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**  $\rightarrow$  [a-zA-Z]
  - digit**  $\rightarrow$  [0-9]
  - id**  $\rightarrow$  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**  $\rightarrow$  digit digit\*
  - optional\_fraction**  $\rightarrow$  . digits |  $\lambda$
  - optional\_exponent**  $\rightarrow$  (E (+ | - |  $\lambda$ ) digits) |  $\lambda$
  - num**  $\rightarrow$  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:    :=