Parser for Compiler in `mosmlyac`

Franklin Chen

July 28, 1996

This is a parser for the project developed in [1], done in `mosmlyac` for Moscow ML rather than in `yacc` for C.

The specification file has a well-defined format:

1a ⟨* 1a⟩≡
```
%{
⟨Header 1b⟩
%}
⟨Declarations 1c⟩
%%
⟨Rules 6b⟩
%%
⟨Trailer 17⟩
```

1 Header

We prefer not to `open` modules unnecessarily.

1b ⟨Header 1b⟩≡ (1a)
```
(* Empty *)
```

2 Declarations

1c ⟨Declarations 1c⟩≡ (1a)
```
⟨Tokens 2⟩
⟨Precedence 5a⟩
⟨Entry points 5b⟩
⟨Other nonterminal types 6a⟩
```

2.1 Tokens

These are the only tokens with attributes.

2 ⟨ *Tokens* 2 ⟩≡ (1c) 3▷

```
%token <string> Identifier
%token <int> Constant
```

Defines:
`Constant`, used in chunk 14a.
`Identifier`, used in chunk 13b.

The rest of the tokens do not need attributes.

3 ⟨ *Tokens* 2⟩+≡ (1c) ◁2 4▷

```
%token INT
%token IF
%token ELSE
%token WHILE
%token BREAK
%token CONTINUE
%token RETURN
%token SEMI
%token LPAREN
%token RPAREN
%token LBRACE
%token RBRACE
%token PLUS
%token MINUS
%token TIMES
%token DIVIDE
%token REM
%token GT
%token LT
%token GE
%token LE
%token EQ
%token NE
%token AMP
%token CARET
%token BAR
%token ASSIGN
%token PE
%token ME
%token TE
%token DE
%token RE
%token PP
%token MM
%token COMMA
```

Defines:
AMP, used in chunks 12a and 15a.
ASSIGN, used in chunks 12a and 15b.
BAR, used in chunks 12a and 15a.
BREAK, used in chunk 13e.
CARET, used in chunks 12a and 15a.
COMMA, used in chunk 13c.
CONTINUE, used in chunk 13e.
DE, used in chunk 15b.
DIVIDE, used in chunk 14b.

ELSE, used in chunk 14a.
EQ, used in chunks 12a and 14c.
GE, used in chunk 14c.
GT, used in chunks 12a and 14c.
IF, used in chunk 14a.
INT, used in chunk 13b.
LBRACE, used in chunk 13e.
LE, used in chunk 14c.
LPAREN, used in chunk 13c.
LT, used in chunk 14c.
ME, used in chunk 15b.
MINUS, used in chunk 14b.
MM, used in chunk 14b.
NE, used in chunk 14c.
PE, used in chunk 15b.
PLUS, used in chunks 12a and 14b.
PP, used in chunks 12a and 14b.
RBRACE, used in chunk 13e.
RE, used in chunk 15b.
REM, used in chunk 14b.
RETURN, used in chunk 13e.
RPAREN, used in chunk 13c.
SEMI, used in chunk 13d.
TE, used in chunk 15b.
TIMES, used in chunks 12a and 14b.
WHILE, used in chunk 14a.

The usual end-of-file token.

4 ⟨ *Tokens* 2⟩+≡ (1c) ◁3

```
%token EOF
```

Defines:
EOF, used in chunk 6c.

Precedences follow, from lowest to highest.

5a ⟨*Precedence* 5a⟩≡ (1c)

```
%right ASSIGN PE ME TE DE RE
%left BAR
%left CARET
%left AMP
%left EQ NE
%left LT GT GE LE
%left PLUS MINUS
%left TIMES DIVIDE REM
%right PP MM
```

Defines:
AMP, used in chunks 12a and 15a.
ASSIGN, used in chunks 12a and 15b.
BAR, used in chunks 12a and 15a.
CARET, used in chunks 12a and 15a.
DE, used in chunk 15b.
DIVIDE, used in chunk 14b.
EQ, used in chunks 12a and 14c.
GE, used in chunk 14c.
GT, used in chunks 12a and 14c.
LE, used in chunk 14c.
LT, used in chunk 14c.
ME, used in chunk 15b.
MINUS, used in chunk 14b.
MM, used in chunk 14b.
NE, used in chunk 14c.
PE, used in chunk 15b.
PLUS, used in chunks 12a and 14b.
PP, used in chunks 12a and 14b.
RE, used in chunk 15b.
REM, used in chunk 14b.
TE, used in chunk 15b.
TIMES, used in chunks 12a and 14b.

2.2 Nonterminals

We only have one entry point.

5b ⟨*Entry points* 5b⟩≡ (1c)

```
%start program
%type <Format.action> program
```

Uses program 6c.

Note the pervasive passing around of `Format.action`.

6a ⟨*Other nonterminal types* 6a⟩≡ (1c)

```
%type <Format.action> definitions
%type <Format.action> definition
%type <Format.action> function_definition
%type <Format.action> optional_parameter_list
%type <Format.action> parameter_list
%type <Format.action> parameter_declarations
%type <Format.action> parameter_declaration
%type <Format.action> parameter_declarator_list
%type <Format.action> compound_statement
%type <Format.action> declarations
%type <Format.action> declaration
%type <Format.action> declarator_list
%type <Format.action> statements
%type <Format.action> statement
%type <Format.action> if_prefix
%type <Format.action> loop_prefix
%type <Format.action> expression
%type <Format.action> binary
%type <Format.action> optional_argument_list
%type <Format.action> argument_list
⟨Dummy nonterminal types 8b⟩
```

Uses `argument_list` 13a, `binary` 12a, `compound_statement` 9c, `declaration` 9f, `declarations` 9e, `declarator_list` 10a, `definition` 7c, `definitions` 7b, `expression` 11b, `function_definition` 8a, `if_prefix` 10e, `loop_prefix` 11a, `optional_argument_list` 12b, `optional_parameter_list` 8c, `parameter_declaration` 9a, `parameter_declarations` 8e, `parameter_declarator_list` 9b, `parameter_list` 8d, `statement` 10c, and `statements` 10b.

3 Rules

Note that various synthesized attributes are now functions, in order to simulate the inherited formatter attribute we pass around to do pretty-printing.

6b ⟨*Rules* 6b⟩≡ (1a)

```
⟨Main entry point 6c⟩
⟨Other rules 7a⟩
```

6c ⟨*Main entry point* 6c⟩≡ (6b)

```
program
        : definitions EOF { $1 }
```

Defines:
`program`, used in chunk 5b.
Uses `definitions` 7b and `EOF` 4.

We will expand each of these later.

7a ⟨*Other rules* 7a⟩≡ (6b)
⟨definitions 7b⟩
⟨definition 7c⟩
⟨function_definition 8a⟩
⟨optional_parameter_list 8c⟩
⟨parameter_list 8d⟩
⟨parameter_declarations 8e⟩
⟨parameter_declaration 9a⟩
⟨parameter_declarator_list 9b⟩
⟨compound_statement 9c⟩
⟨declarations 9e⟩
⟨declaration 9f⟩
⟨declarator_list 10a⟩
⟨statements 10b⟩
⟨statement 10c⟩
⟨if_prefix 10e⟩
⟨loop_prefix 11a⟩
⟨expression 11b⟩
⟨binary 12a⟩
⟨optional_argument_list 12b⟩
⟨argument_list 13a⟩
⟨*Rules for dummy nonterminals* 13b⟩

Use function composition to build up a composite action that will be executed.

7b ⟨definitions 7b⟩≡ (7a)
```
definitions
        : definition { $1 }
        | definitions definition { $2 o $1 }
```
Defines:
definitions, used in chunk 6.
Uses definition 7c.

We replace the former INT with a simple nonterminal n_int in order to associate actions with tokens.

7c ⟨definition 7c⟩≡ (7a)
```
definition
        : function_definition { $1 }
        | n_int function_definition { $2 o $1 }
        | declaration { $1 }
```
Defines:
definition, used in chunks 6a and 7b.
Uses declaration 9f, function_definition 8a, and n_int 13b.

Note that function_definition_1 and function_definition_2 had to be introduced because unlike yacc, mosmlyac does not support actions within rules.

8a ⟨function_definition 8a⟩≡ (7a)

```
function_definition
        : n_identifier n_lp optional_parameter_list n_rp
          function_definition_1
          parameter_declarations
          function_definition_2
          compound_statement { (Format.nl Format.AT) o
                               $8 o $7 o $6 o $5 o $4 o $3 o $2 o $1 }

function_definition_1: { Format.nl Format.IN }

function_definition_2: { Format.at Format.EX }
```

Defines:
function_definition, used in chunks 6a and 7c.
function_definition_1, used in chunk 8b.
function_definition_2, used in chunk 8b.
Uses compound_statement 9c, n_identifier 13b, n_lp 13c, n_rp 13c, optional_parameter_list 8c, and parameter_declarations 8e.

8b ⟨*Dummy nonterminal types* 8b⟩≡ (6a) 9d▷

```
%type <Format.action> function_definition_1
%type <Format.action> function_definition_2
```

Uses function_definition_1 8a and function_definition_2 8a.

8c ⟨optional_parameter_list 8c⟩≡ (7a)

```
optional_parameter_list
        : /* no formal parameters */ { Format.unitAction }
        | parameter_list { $1 }
```

Defines:
optional_parameter_list, used in chunks 6a and 8a.
Uses parameter_list 8d.

8d ⟨parameter_list 8d⟩≡ (7a)

```
parameter_list
        : n_identifier { $1 }
        | parameter_list n_co n_identifier { $3 o $2 o $1 }
```

Defines:
parameter_list, used in chunks 6a and 8c.
Uses n_co 13c and n_identifier 13b.

8e ⟨parameter_declarations 8e⟩≡ (7a)

```
parameter_declarations
        : /* null */ { Format.unitAction }
        | parameter_declarations parameter_declaration { $2 o $1 }
```

Defines:
parameter_declarations, used in chunks 6a and 8a.
Uses parameter_declaration 9a.

9a ⟨parameter_declaration 9a⟩≡ (7a)

```
parameter_declaration
        : n_int parameter_declarator_list n_sc { $3 o $2 o $1 }
```

Defines:
parameter_declaration, used in chunks 6a and 8e.
Uses n_int 13b and parameter_declarator_list 9b.

9b ⟨parameter_declarator_list 9b⟩≡ (7a)

```
parameter_declarator_list
        : n_identifier { $1 }
        | parameter_declarator_list n_co n_identifier { $3 o $2 o $1 }
```

Defines:
parameter_declarator_list, used in chunks 6a and 9a.
Uses n_co 13c and n_identifier 13b.

9c ⟨compound_statement 9c⟩≡ (7a)

```
compound_statement
        : n_lr declarations
          compound_statement_1
          statements n_rr { $5 o $4 o $3 o $2 o $1 }

compound_statement_1: { Format.nl Format.AT }
```

Defines:
compound_statement, used in chunks 6a, 8a, and 10c.
compound_statement_1, used in chunk 9d.
Uses declarations 9e, n_lr 13e, n_rr 13e, and statements 10b.

9d ⟨*Dummy nonterminal types* 8b⟩+≡ (6a) ◁8b 10d▷

```
%type <Format.action> compound_statement_1
```

Uses compound_statement_1 9c.

9e ⟨declarations 9e⟩≡ (7a)

```
declarations
        : /* null */ { Format.unitAction }
        | declarations declaration { $2 o $1 }
```

Defines:
declarations, used in chunks 6a and 9c.
Uses declaration 9f.

9f ⟨declaration 9f⟩≡ (7a)

```
declaration
        : n_int declarator_list n_sc { $3 o $2 o $1 }
```

Defines:
declaration, used in chunks 6a, 7c, and 9e.
Uses declarator_list 10a and n_int 13b.

10a ⟨declarator_list 10a⟩≡ (7a)

```
declarator_list
        : n_identifier { $1 }
        | declarator_list n_co n_identifier { $3 o $2 o $1 }
```

Defines:
declarator_list, used in chunks 6a and 9f.
Uses n_co 13c and n_identifier 13b.

10b ⟨statements 10b⟩≡ (7a)

```
statements
        : /* null */ { Format.unitAction }
        | statements statement { $2 o $1 }
```

Defines:
statements, used in chunks 6a and 9c.
Uses statement 10c.

10c ⟨statement 10c⟩≡ (7a)

```
statement
        : expression n_sc { $2 o $1 }
        | n_sc  /* null statement */ { $1 }
        | n_break n_sc { $2 o $1 }
        | n_continue n_sc { $2 o $1 }
        | n_return n_sc { $2 o $1 }
        | n_return
          statement_1
          expression n_sc { $4 o $3 o $2 o $1 }
        | compound_statement { $1 }
        | if_prefix statement { (Format.uncond Format.EX) o $2 o $1 }
        | if_prefix statement n_else statement { (Format.uncond Format.EX) o
                                                  $4 o $3 o $2 o $1}
        | loop_prefix statement { (Format.uncond Format.EX) o $2 o $1 }

statement_1: { Format.out " " }
```

Defines:
statement, used in chunks 6a and 10b.
statement_1, used in chunk 10d.
Uses compound_statement 9c, expression 11b, if_prefix 10e, loop_prefix 11a, n_break 13e, n_continue 13e, n_else 14a, and n_return 13e.

10d ⟨*Dummy nonterminal types* 8b⟩+≡ (6a) ◁9d 16▷

```
%type <Format.action> statement_1
```

Uses statement_1 10c.

10e ⟨if_prefix 10e⟩≡ (7a)

```
if_prefix
        : n_if n_lp expression n_rp { (Format.cond Format.IN) o
                                      $4 o $3 o $2 o $1 }
```

Defines:
if_prefix, used in chunks 6a and 10c.
Uses expression 11b, n_if 14a, n_lp 13c, and n_rp 13c.

11a ⟨loop_prefix 11a⟩≡ (7a)

```
loop_prefix
        : n_while n_lp expression n_rp { (Format.cond Format.IN) o
                                         $4 o $3 o $2 o $1 }
```

Defines:
loop_prefix, used in chunks 6a and 10c.
Uses expression 11b, n_lp 13c, n_rp 13c, and n_while 14a.

11b ⟨expression 11b⟩≡ (7a)

```
expression
        : binary { $1 }
        | expression n_co binary { $3 o $2 o $1 }
```

Defines:
expression, used in chunks 6a and 10–12.
Uses binary 12a and n_co 13c.

12a ⟨binary 12a⟩≡ (7a)

```
binary
        : n_identifier { $1 }
        | n_constant { $1 }
        | n_lp expression n_rp { $3 o $2 o $1 }
        | n_identifier n_lp optional_argument_list n_rp { $4 o $3 o $2 o $1 }
        | n_pp n_identifier %prec PP { $2 o $1 }
        | n_mm n_identifier %prec PP { $2 o $1 }
        | binary n_pl binary %prec PLUS { $3 o $2 o $1 }
        | binary n_mi binary %prec PLUS { $3 o $2 o $1 }
        | binary n_mu binary %prec TIMES { $3 o $2 o $1 }
        | binary n_di binary %prec TIMES { $3 o $2 o $1 }
        | binary n_rm binary %prec TIMES { $3 o $2 o $1 }
        | binary n_gt binary %prec GT { $3 o $2 o $1 }
        | binary n_lt binary %prec GT { $3 o $2 o $1 }
        | binary n_ge binary %prec GT { $3 o $2 o $1 }
        | binary n_le binary %prec GT { $3 o $2 o $1 }
        | binary n_eq binary %prec EQ { $3 o $2 o $1 }
        | binary n_ne binary %prec EQ { $3 o $2 o $1 }
        | binary n_an binary %prec AMP { $3 o $2 o $1 }
        | binary n_xo binary %prec CARET { $3 o $2 o $1 }
        | binary n_or binary %prec BAR { $3 o $2 o $1 }
        | n_identifier n_as binary %prec ASSIGN { $3 o $2 o $1 }
        | n_identifier n_pe binary %prec ASSIGN { $3 o $2 o $1 }
        | n_identifier n_me binary %prec ASSIGN { $3 o $2 o $1 }
        | n_identifier n_te binary %prec ASSIGN { $3 o $2 o $1 }
        | n_identifier n_de binary %prec ASSIGN { $3 o $2 o $1 }
        | n_identifier n_re binary %prec ASSIGN { $3 o $2 o $1 }
```

Defines:
binary, used in chunks 6a, 11b, and 13a.
Uses AMP 3 5a, ASSIGN 3 5a, BAR 3 5a, CARET 3 5a, EQ 3 5a, expression 11b, GT 3 5a, n_an 15a, n_as 15b, n_constant 14a, n_de 15b, n_di 14b, n_eq 14c, n_ge 14c, n_gt 14c, n_identifier 13b, n_le 14c, n_lp 13c, n_lt 14c, n_me 15b, n_mi 14b, n_mm 14b, n_mu 14b, n_ne 14c, n_or 15a, n_pe 15b, n_pl 14b, n_pp 14b, n_re 15b, n_rm 14b, n_rp 13c, n_te 15b, n_xo 15a, optional_argument_list 12b, PLUS 3 5a, PP 3 5a, and TIMES 3 5a.

12b ⟨optional_argument_list 12b⟩≡ (7a)

```
optional_argument_list
        : /* no actual arguments */ { Format.unitAction }
        | argument_list { $1 }
```

Defines:
optional_argument_list, used in chunks 6a and 12a.
Uses argument_list 13a.

13a ⟨argument_list 13a⟩≡ (7a)
```
argument_list
        : binary { $1 }
        | argument_list n_co binary { $3 o $2 o $1 }
```
Defines:
argument_list, used in chunks 6a and 12b.
Uses binary 12a and n_co 13c.

3.1 Dummy nonterminals

These nonterminals are used to cause actions to be performed for tokens.

13b ⟨*Rules for dummy nonterminals* 13b⟩≡ (7a) 13c▷
```
n_int: INT { Format.out "int\t" }
n_identifier: Identifier { Format.out $1 }
```
Defines:
n_identifier, used in chunks 8–10, 12a, and 16.
n_int, used in chunks 7c, 9, and 16.
Uses Identifier 2 and INT 3.

13c ⟨*Rules for dummy nonterminals* 13b⟩+≡ (7a) ◁13b 13d▷
```
n_lp: LPAREN { Format.out "(" }
n_rp: RPAREN { Format.out ")" }
n_co: COMMA { Format.out ", " }
```
Defines:
n_co, used in chunks 8–11, 13a, and 16.
n_lp, used in chunks 8a, 10–12, and 16.
n_rp, used in chunks 8a, 10–12, and 16.
Uses COMMA 3, LPAREN 3, and RPAREN 3.

13d ⟨*Rules for dummy nonterminals* 13b⟩+≡ (7a) ◁13c 13e▷
```
n_sc: SEMI { (Format.nl Format.AT) o (Format.out ";") }
```
Uses SEMI 3.

13e ⟨*Rules for dummy nonterminals* 13b⟩+≡ (7a) ◁13d 14a▷
```
n_break: BREAK { Format.out "break" }
n_continue: CONTINUE { Format.out "continue" }
n_return: RETURN { Format.out "return" }
n_lr: LBRACE { (Format.at Format.IN) o
  (Format.out "{\t") o (Format.cond Format.EX) }
n_rr: RBRACE { (Format.uncond Format.AT) o
  (Format.out "}") o (Format.at Format.EX) }
```
Defines:
n_break, used in chunks 10c and 16.
n_continue, used in chunks 10c and 16.
n_lr, used in chunks 9c and 16.
n_return, used in chunks 10c and 16.
n_rr, used in chunks 9c and 16.
Uses BREAK 3, CONTINUE 3, LBRACE 3, RBRACE 3, and RETURN 3.

14a ⟨*Rules for dummy nonterminals* 13b⟩+≡ (7a) ◁13e 14b▷

```
n_if: IF { Format.out "if " }
n_else: ELSE { (Format.cond Format.IN) o
  (Format.out "else") o (Format.at Format.EX) }
n_while: WHILE { Format.out "while " }
n_constant: Constant { Format.out (Int.toString $1) }
```

Defines:
n_constant, used in chunks 12a and 16.
n_else, used in chunks 10c and 16.
n_if, used in chunks 10e and 16.
n_while, used in chunks 11a and 16.
Uses Constant 2, ELSE 3, IF 3, and WHILE 3.

14b ⟨*Rules for dummy nonterminals* 13b⟩+≡ (7a) ◁14a 14c▷

```
n_pp: PP { Format.out " ++ " }
n_mm: MM { Format.out " -- " }
n_pl: PLUS { Format.out " + " }
n_mi: MINUS { Format.out " - " }
n_mu: TIMES { Format.out " * " }
n_di: DIVIDE { Format.out " / " }
n_rm: REM { Format.out " % " }
```

Defines:
n_di, used in chunks 12a and 16.
n_mi, used in chunks 12a and 16.
n_mm, used in chunks 12a and 16.
n_mu, used in chunks 12a and 16.
n_pl, used in chunks 12a and 16.
n_pp, used in chunks 12a and 16.
n_rm, used in chunks 12a and 16.
Uses DIVIDE 3 5a, MINUS 3 5a, MM 3 5a, PLUS 3 5a, PP 3 5a, REM 3 5a, and TIMES 3 5a.

14c ⟨*Rules for dummy nonterminals* 13b⟩+≡ (7a) ◁14b 15a▷

```
n_gt: GT { Format.out " > " }
n_lt: LT { Format.out " < " }
n_ge: GE { Format.out " >= " }
n_le: LE { Format.out " <= " }
n_eq: EQ { Format.out " == " }
n_ne: NE { Format.out " != " }
```

Defines:
n_eq, used in chunks 12a and 16.
n_ge, used in chunks 12a and 16.
n_gt, used in chunks 12a and 16.
n_le, used in chunks 12a and 16.
n_lt, used in chunks 12a and 16.
n_ne, used in chunks 12a and 16.
Uses EQ 3 5a, GE 3 5a, GT 3 5a, LE 3 5a, LT 3 5a, and NE 3 5a.

15a ⟨*Rules for dummy nonterminals* 13b⟩+≡ (7a) ◁14c 15b▷

```
n_an: AMP { Format.out " & " }
n_xo: CARET { Format.out " ^ " }
n_or: BAR { Format.out " | " }
```

Defines:
n_an, used in chunks 12a and 16.
n_or, used in chunks 12a and 16.
n_xo, used in chunks 12a and 16.
Uses AMP 3 5a, BAR 3 5a, and CARET 3 5a.

15b ⟨*Rules for dummy nonterminals* 13b⟩+≡ (7a) ◁15a

```
n_as: ASSIGN { Format.out " = " }
n_pe: PE { Format.out " += " }
n_me: ME { Format.out " -= " }
n_te: TE { Format.out " *= " }
n_de: DE { Format.out " /= " }
n_re: RE { Format.out " %= " }
```

Defines:
n_as, used in chunks 12a and 16.
n_de, used in chunks 12a and 16.
n_me, used in chunks 12a and 16.
n_pe, used in chunks 12a and 16.
n_re, used in chunks 12a and 16.
n_te, used in chunks 12a and 16.
Uses ASSIGN 3 5a, DE 3 5a, ME 3 5a, PE 3 5a, RE 3 5a, and TE 3 5a.

16 ⟨*Dummy nonterminal types* 8b⟩+≡ (6a) ◁10d

```
%type <Format.action> n_int
%type <Format.action> n_identifier
%type <Format.action> n_lp
%type <Format.action> n_rp
%type <Format.action> n_co
%type <Format.action> n_sc
%type <Format.action> n_break
%type <Format.action> n_continue
%type <Format.action> n_return
%type <Format.action> n_lr
%type <Format.action> n_rr
%type <Format.action> n_if
%type <Format.action> n_else
%type <Format.action> n_while
%type <Format.action> n_constant
%type <Format.action> n_pp
%type <Format.action> n_mm
%type <Format.action> n_pl
%type <Format.action> n_mi
%type <Format.action> n_mu
%type <Format.action> n_di
%type <Format.action> n_rm
%type <Format.action> n_gt
%type <Format.action> n_lt
%type <Format.action> n_ge
%type <Format.action> n_le
%type <Format.action> n_eq
%type <Format.action> n_ne
%type <Format.action> n_an
%type <Format.action> n_xo
%type <Format.action> n_or
%type <Format.action> n_as
%type <Format.action> n_pe
%type <Format.action> n_me
%type <Format.action> n_te
%type <Format.action> n_de
%type <Format.action> n_re
```

Uses n_an 15a, n_as 15b, n_break 13e, n_co 13c, n_constant 14a, n_continue 13e, n_de 15b, n_di 14b, n_else 14a, n_eq 14c, n_ge 14c, n_gt 14c, n_identifier 13b, n_if 14a, n_int 13b, n_le 14c, n_lp 13c, n_lr 13e, n_lt 14c, n_me 15b, n_mi 14b, n_mm 14b, n_mu 14b, n_ne 14c, n_or 15a, n_pe 15b, n_pl 14b, n_pp 14b, n_re 15b, n_return 13e, n_rm 14b, n_rp 13c, n_rr 13e, n_te 15b, n_while 14a, and n_xo 15a.

4 Trailer

Empty trailer.

17 ⟨ *Trailer* 17⟩≡ (1a)

5 Limitations

All the token types should really be augmented with position information passed by the lexer, in order to be able to generate informative error messages.

6 Indices

6.1 Chunks

⟨* 1a⟩
⟨argument_list 13a⟩
⟨binary 12a⟩
⟨compound_statement 9c⟩
⟨declaration 9f⟩
⟨declarations 9e⟩
⟨declarator_list 10a⟩
⟨definition 7c⟩
⟨definitions 7b⟩
⟨expression 11b⟩
⟨function_definition 8a⟩
⟨if_prefix 10e⟩
⟨loop_prefix 11a⟩
⟨optional_argument_list 12b⟩
⟨optional_parameter_list 8c⟩
⟨parameter_declaration 9a⟩
⟨parameter_declarations 8e⟩
⟨parameter_declarator_list 9b⟩
⟨parameter_list 8d⟩
⟨statement 10c⟩
⟨statements 10b⟩
⟨*Declarations* 1c⟩
⟨*Dummy nonterminal types* 8b⟩
⟨*Entry points* 5b⟩
⟨*Header* 1b⟩
⟨*Main entry point* 6c⟩
⟨*Other nonterminal types* 6a⟩
⟨*Other rules* 7a⟩
⟨*Precedence* 5a⟩
⟨*Rules* 6b⟩
⟨*Rules for dummy nonterminals* 13b⟩
⟨*Tokens* 2⟩
⟨*Trailer* 17⟩

6.2 Identifiers

AMP: 3, 5a, 12a, 15a
argument_list: 6a, 12b, 13a
ASSIGN: 3, 5a, 12a, 15b
BAR: 3, 5a, 12a, 15a
binary: 6a, 11b, 12a, 13a
BREAK: 3, 13e
CARET: 3, 5a, 12a, 15a
COMMA: 3, 13c
compound_statement: 6a, 8a, 9c, 10c
compound_statement_1: 9c, 9d
Constant: 2, 14a
CONTINUE: 3, 13e
DE: 3, 5a, 15b
declaration: 6a, 7c, 9e, 9f
declarations: 6a, 9c, 9e
declarator_list: 6a, 9f, 10a
definition: 6a, 7b, 7c
definitions: 6a, 6c, 7b
DIVIDE: 3, 5a, 14b
ELSE: 3, 14a
EOF: 4, 6c
EQ: 3, 5a, 12a, 14c
expression: 6a, 10c, 10e, 11a, 11b, 12a
function_definition: 6a, 7c, 8a
function_definition_1: 8a, 8b
function_definition_2: 8a, 8b
GE: 3, 5a, 14c
GT: 3, 5a, 12a, 14c
Identifier: 2, 13b
IF: 3, 14a
if_prefix: 6a, 10c, 10e
INT: 3, 13b
LBRACE: 3, 13e
LE: 3, 5a, 14c
loop_prefix: 6a, 10c, 11a
LPAREN: 3, 13c
LT: 3, 5a, 14c
ME: 3, 5a, 15b
MINUS: 3, 5a, 14b
MM: 3, 5a, 14b
n_an: 12a, 15a, 16
n_as: 12a, 15b, 16
n_break: 10c, 13e, 16
n_co: 8d, 9b, 10a, 11b, 13a, 13c, 16

n_constant: 12a, 14a, 16
n_continue: 10c, 13e, 16
n_de: 12a, 15b, 16
n_di: 12a, 14b, 16
n_else: 10c, 14a, 16
n_eq: 12a, 14c, 16
n_ge: 12a, 14c, 16
n_gt: 12a, 14c, 16
n_identifier: 8a, 8d, 9b, 10a, 12a, 13b, 16
n_if: 10e, 14a, 16
n_int: 7c, 9a, 9f, 13b, 16
n_le: 12a, 14c, 16
n_lp: 8a, 10e, 11a, 12a, 13c, 16
n_lr: 9c, 13e, 16
n_lt: 12a, 14c, 16
n_me: 12a, 15b, 16
n_mi: 12a, 14b, 16
n_mm: 12a, 14b, 16
n_mu: 12a, 14b, 16
n_ne: 12a, 14c, 16
n_or: 12a, 15a, 16
n_pe: 12a, 15b, 16
n_pl: 12a, 14b, 16
n_pp: 12a, 14b, 16
n_re: 12a, 15b, 16
n_return: 10c, 13e, 16
n_rm: 12a, 14b, 16
n_rp: 8a, 10e, 11a, 12a, 13c, 16
n_rr: 9c, 13e, 16
n_te: 12a, 15b, 16
n_while: 11a, 14a, 16
n_xo: 12a, 15a, 16
NE: 3, 5a, 14c
optional_argument_list: 6a, 12a, 12b
optional_parameter_list: 6a, 8a, 8c
parameter_declaration: 6a, 8e, 9a
parameter_declarations: 6a, 8a, 8e
parameter_declarator_list: 6a, 9a, 9b
parameter_list: 6a, 8c, 8d
PE: 3, 5a, 15b
PLUS: 3, 5a, 12a, 14b
PP: 3, 5a, 12a, 14b
program: 5b, 6c
RBRACE: 3, 13e
RE: 3, 5a, 15b
REM: 3, 5a, 14b

RETURN: 3, 13e
RPAREN: 3, 13c
SEMI: 3, 13d
statement: 6a, 10b, 10c
statement_1: 10c, 10d
statements: 6a, 9c, 10b
TE: 3, 5a, 15b
TIMES: 3, 5a, 12a, 14b
WHILE: 3, 14a

References

[1] Axel T. Schreiner and H. George Friedman, Jr. *Introduction to Compiler Construction with UNIX*[1] . Prentice-Hall, Inc., New Jersey, 1985.

[1] UNIX is a trademark of Bell Laboratories.