%token IDENTIFIER CONSTANT STRING\_LITERAL TYPE MAIN

%token READ WRITE

%token INC\_OP DEC\_OP LEFT\_OP RIGHT\_OP LE\_OP GE\_OP EQ\_OP NE\_OP

%token AND\_OP OR\_OP MUL\_ASSIGN DIV\_ASSIGN MOD\_ASSIGN ADD\_ASSIGN

%token SUB\_ASSIGN LEFT\_ASSIGN RIGHT\_ASSIGN AND\_ASSIGN

%token XOR\_ASSIGN OR\_ASSIGN

%token IF ELSE WHILE DO FOR CONTINUE BREAK RETURN

%start s

%%

s

: MAIN compound\_statement

;

compound\_statement

: '{' '}'

| '{' statement\_list '}'

| '{' declaration\_list '}'

| '{' declaration\_list statement\_list '}'

;

declaration\_list

: declaration

| declaration\_list declaration

;

statement\_list

: statement

| statement\_list statement

;

statement

: compound\_statement

| expression\_statement

| selection\_statement

| iteration\_statement

| jump\_statement

| read\_write\_statement

;

declaration

: TYPE init\_declarator\_list ';'

;

init\_declarator\_list

: init\_declarator

| init\_declarator\_list ',' init\_declarator

;

init\_declarator

: declarator

| declarator '=' initializer

;

declarator

: IDENTIFIER

| declarator '[' expression ']'

| declarator '[' ']'

;

initializer

: assignment\_expression

| '{' initializer\_list '}'

;

initializer\_list

: initializer

| initializer\_list ',' initializer

;

expression\_statement

: ';'

| expression ';'

;

read\_write\_statement

: READ '(' expression ')' ';'

| WRITE '(' expression ')' ';'

;

expression

: assignment\_expression

| expression ',' assignment\_expression

;

selection\_statement

: IF '(' expression ')' statement

| IF '(' expression ')' statement ELSE statement

;

iteration\_statement

: while\_statement

| do\_while\_statement

| for\_statement

;

while\_statement

: WHILE '(' expression ')' statement

;

do\_while\_statement

DO statement WHILE '(' expression ')' ';'

;

for\_statement

FOR '(' expression\_statement expression\_statement ')' statement

| FOR '(' expression\_statement expression\_statement expression ')' statement

;

jump\_statement

: CONTINUE ';'

| BREAK ';'

| RETURN ';'

| RETURN expression ';'

;

assignment\_expression

: logical\_or\_expression

| unary\_expression assignment\_operator assignment\_expression

;

assignment\_operator

: '='

| MUL\_ASSIGN

| DIV\_ASSIGN

| MOD\_ASSIGN

| ADD\_ASSIGN

| SUB\_ASSIGN

| LEFT\_ASSIGN

| RIGHT\_ASSIGN

| AND\_ASSIGN

| XOR\_ASSIGN

| OR\_ASSIGN

;

unary\_expression

: postfix\_expression

| unary\_operator unary\_expression

| INC\_OP unary\_expression

| DEC\_OP unary\_expression

;

unary\_operator

: '-'

| '!'

;

postfix\_expression

: primary\_expression

| postfix\_expression '[' expression ']'

| postfix\_expression '(' ')'

| postfix\_expression '(' argument\_expression\_list ')'

| postfix\_expression INC\_OP

| postfix\_expression DEC\_OP

;

argument\_expression\_list

: assignment\_expression

| argument\_expression\_list ',' assignment\_expression

;

multiplicative\_expression

: unary\_expression

| multiplicative\_expression '\*' unary\_expression

| multiplicative\_expression '/' unary\_expression

| multiplicative\_expression '%' unary\_expression

;

additive\_expression

: multiplicative\_expression

| additive\_expression '+' multiplicative\_expression

| additive\_expression '-' multiplicative\_expression

;

shift\_expression

: additive\_expression

| shift\_expression LEFT\_OP additive\_expression

| shift\_expression RIGHT\_OP additive\_expression

;

relational\_expression

: shift\_expression

| relational\_expression '<' shift\_expression

| relational\_expression '>' shift\_expression

| relational\_expression LE\_OP shift\_expression

| relational\_expression GE\_OP shift\_expression

;

equality\_expression

: relational\_expression

| equality\_expression EQ\_OP relational\_expression

| equality\_expression NE\_OP relational\_expression

;

and\_expression

: equality\_expression

| and\_expression '&' equality\_expression

;

exclusive\_or\_expression

: and\_expression

| exclusive\_or\_expression '^' and\_expression

;

inclusive\_or\_expression

: exclusive\_or\_expression

| inclusive\_or\_expression '|' exclusive\_or\_expression

;

logical\_and\_expression

: inclusive\_or\_expression

| logical\_and\_expression AND\_OP inclusive\_or\_expression

;

logical\_or\_expression

: logical\_and\_expression

| logical\_and\_expression OR\_OP logical\_or\_expression

;

primary\_expression

: IDENTIFIER

| CONSTANT

| STRING\_LITERAL

| '(' expression ')'

;

%%