EBNF

```
program : translation_unit;

abstract_declarator
                : pointer direct_abstract_declarator
                | pointer
                | direct_abstract_declarator
                ;
argument_expression_list
                : assignment_expression
                | argument_expression_list ',' assignment_expression
                ;
assignment_operator
                : '='
                ;
block_item_list
                : block_item
                | block_item_list block_item
                ;
block_item
                : declaration
                | statement
                ;
compound_statement

                : '{' '}'
                | '{'  block_item_list '}'
                ;
constant
                : INT_C
                | FLOAT_C
                ;
declaration
                : declaration_specifiers ';'
                | declaration_specifiers init_declarator_list ';'
                ;
declarator
                : pointer direct_
                | direct_declarator
                ;
declaration_specifiers
                : storage_class_specifier declaration_specifiers
                | storage_class_specifier
                | type_specifier declaration_specifiers
                | type_specifier
                | type_qualifier declaration_specifiers
                | type_qualifier
                ;
declaration_list
                : declaration
                | declaration_list declaration
                ;
direct_declarator
                : IDENTIFIER
                | '(' declarator ')'
                | direct_declarator '[' ']'
                | direct_declarator '[' '*' ']'
                | direct_declarator '[' 🌱type_qualifier_list assignment_expression ']'
                | direct_declarator '[' 🌱assignment_expression ']'
                | direct_declarator '[' type_qualifier_list '*' ']'
                | direct_declarator '[' type_qualifier_list 🌱assignment_expression ']'
                | direct_declarator '[' type_qualifier_list assignment_expression ']'
                | direct_declarator '[' type_qualifier_list ']'
                | direct_declarator '[' assignment_expression ']'
                | direct_declarator '(' parameter_type_list ')'
                | direct_declarator '(' ')'
                | direct_declarator '(' identifier_list ')'
                ;
designation
                : designator_list '='
                ;
designator_list
                : designator
                | designator_list designator
                ;
designator
                : '[' constant_expression ']'
                | '.' IDENTIFIER
                ;
direct_abstract_declarator
                : '(' abstract_declarator ')'
                | '[' ']'
                | '[' '*' ']'
                | '[' ⚘ type_qualifier_list assignment_expression ']'
                | '[' ⚘ assignment_expression ']'
                | '[' type_qualifier_list ⚘ assignment_expression ']'
                | '[' type_qualifier_list assignment_expression ']'
                | '[' type_qualifier_list ']'
                | '[' assignment_expression ']'
                | direct_abstract_declarator '[' ']'
                | direct_abstract_declarator '[' '*' ']'
                | direct_abstract_declarator '[' ⚘ type_qualifier_list assignment_expression ']'
                | direct_abstract_declarator '[' ⚘ assignment_expression ']'
                | direct_abstract_declarator '[' type_qualifier_list assignment_expression ']'
                | direct_abstract_declarator '[' type_qualifier_list ⚘ assignment_expression ']'
                | direct_abstract_declarator '[' type_qualifier_list ']'
                | direct_abstract_declarator '[' assignment_expression ']'
                | '(' ')'
                | '(' parameter_type_list ')'
                | direct_abstract_declarator '(' ')'
                | direct_abstract_declarator '(' parameter_type_list ')'
                ;
external_declaration
                : function_definition
                | declaration
                ;
expression_statement
                : ';'
                | expression ';'
                ;
function_definition
                : declaration_specifiers declarator declaration_list compound_statement
                | declaration_specifiers declarator compound_statement
                ;
identifier_list
                : IDENTIFIER
                | identifier_list ',' IDENTIFIER
                ;
initializer
                : '{' initializer_list '}'
                | '{' initializer_list ',' '}'
                | assignment_expression
                 ;
initializer_list
                : designation initializer
                | initializer
                | initializer_list ',' designation initializer
                | initializer_list ',' initializer
                ;
```

```
init_declarator
                : declarator '=' initializer
                | declarator
                ;
init_declarator_list
                : init_declarator
                | init_declarator_list ',' init_declarator
                ;
iteration_statement
                : 🔁  (' expression ')' statement
                | 👉statement 🔁'(' expression ')' ';'
                | 🔂'(' expression_statement expression_statement ')' statement
                | 🔂'(' expression_statement expression_statement expression ')' statement
                | 🔂'(' declaration expression_statement ')' statement
                | 🔂'(' declaration expression_statement expression ')' statement
                ;
jump_statement
                : ➡';'
                | 💔';'
                | 📤';'
                | 📤 expression ';'
                ;
labeled_statement
                : IDENTIFIER ':' statement
                | 💼 constant_expression ':' statement
                ;
parameter_type_list
                : parameter_list ',' ELLIPSIS
                | parameter_list
                ;
parameter_list
                : parameter_declaration
                | parameter_list ',' parameter_declaration
                ;
parameter_declaration
                : declaration_specifiers declarator
                | declaration_specifiers abstract_declarator
                | declaration_specifiers
                ;
pointer
                : '*' type_qualifier_list pointer
                | '*' type_qualifier_list
                | '*' pointer
                | '*'
                ;
selection_statement
                : 🤔 '(' expression ')' statement 🙄 statement
                | 🤔'(' expression ')' statement
                ;
specifier_qualifier_list
                : type_specifier specifier_qualifier_list
                | type_specifier
                | type_qualifier specifier_qualifier_list
                | type_qualifier
                ;
statement
                : labeled_statement
                | compound_statement
                | expression_statement
                | selection_statement
                | iteration_statement
                | jump_statement
                ;
string
    : STRING_LITERAL
    | FUNC_NAME
    ;
storage_class_specifier
                    : ◎
                | 🌱
                ;
type_name
                : specifier_qualifier_list abstract_declarator
                | specifier_qualifier_list
                ;
translation_unit
                : external_declaration
                | translation_unit external_declaration
                ;
type_qualifier
                : ©
                ;
type_qualifier_list
                : type_qualifier
                | type_qualifier_list type_qualifier
                ;
type_specifier
                : VOID
                | CHAR
                | INT
                | LONG
                | DOUBLE
                | BOOL
                ;
unary_operator
                : '&'
                | '*'
                | '+'
                | '-'
                | '~'
                | '!'
                ;
primary_expression
                : IDENTIFIER
                | constant
                | string
                | '(' expression ')'
                ;
postfix_expression
                : primary_expression
                | postfix_expression '[' expression ']'
                | postfix_expression '(' ')'
                | postfix_expression '(' argument_expression_list ')'
                | postfix_expression '.' IDENTIFIER
                | postfix_expression PTR_OP IDENTIFIER
                | postfix_expression INC_OP
                | postfix_expression DEC_OP
                | '(' type_name ')' '{' initializer_list '}'
                | '(' type_name ')' '{' initializer_list ',' '}'
                ;
unary_expression
                : postfix_expression
                | INC_OP unary_expression
                | DEC_OP unary_expression
                | unary_operator cast_expression
                ;
cast_expression
                : unary_expression
                | '(' type_name ')' cast_expression
                ;
```

```
multiplicative_expression
                : cast_expression
                | multiplicative_expression '*' cast_expression
                | multiplicative_expression '/' cast_expression
                | multiplicative_expression '%' cast_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_or_expression OR_OP logical_and_expression
                ;
conditional_expression
                : logical_or_expression
                ;
assignment_expression
                : conditional_expression
                | unary_expression assignment_operator assignment_expression
                ;
constant_expression
                : conditional_expression
                ;
expression
                : assignment_expression
                | expression ',' assignment_expression
                ;
```