

ANSI-C Yacc grammar

```

primary_expression
: IDENTIFIER
| CONSTANT
| STRING_LITERAL
| '(' 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
;

argument_expression_list
: assignment_expression
| argument_expression_list ',' assignment_expression
;

unary_expression
: postfix_expression
| INC_OP unary_expression
| DEC_OP unary_expression
| unary_operator cast_expression
| SIZEOF unary_expression
| SIZEOF '(' type_name ')'
;

unary_operator
: '&'
| '*'
| '+'
| '-'
| '~'
| '!'
;

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
| logical_or_expression '?' expression ':' conditional_expression
;

assignment_expression
: conditional_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
;

expression
: assignment_expression
| expression ',' assignment_expression
;

constant_expression
: conditional_expression
;

declaration
: declaration_specifiers ';'
| declaration_specifiers init_declarator_list ';'

```

```

;
declaration_specifiers
: storage_class_specifier
| storage_class_specifier declaration_specifiers
| type_specifier
| type_specifier declaration_specifiers
| type_qualifier
| type_qualifier declaration_specifiers
;

init_declarator_list
: init_declarator
| init_declarator_list ',' init_declarator
;

init_declarator
: declarator
| declarator '=' initializer
;

storage_class_specifier
: TYPEDEF
| EXTERN
| STATIC
| AUTO
| REGISTER
;

type_specifier
: VOID
| CHAR
| SHORT
| INT
| LONG
| FLOAT
| DOUBLE
| SIGNED
| UNSIGNED
| struct_or_union_specifier
| enum_specifier
| TYPE_NAME
;

struct_or_union_specifier
: struct_or_union IDENTIFIER '{' struct_declaration_list '}'
| struct_or_union '{' struct_declaration_list '}'
| struct_or_union IDENTIFIER
;

struct_or_union
: STRUCT
| UNION
;

struct_declaration_list
: struct_declaration
| struct_declaration_list struct_declaration
;

struct_declaration
: specifier_qualifier_list struct_declarator_list ';'
;

specifier_qualifier_list
: type_specifier specifier_qualifier_list
| type_specifier
| type_qualifier specifier_qualifier_list
| type_qualifier
;

struct_declarator_list
: struct_declarator
| struct_declarator_list ',' struct_declarator
;

struct_declarator
: declarator
| ':' constant_expression
| declarator ':' constant_expression
;

enum_specifier
: ENUM '{' enumerator_list '}'
| ENUM IDENTIFIER '{' enumerator_list '}'
| ENUM IDENTIFIER
;

enumerator_list
: enumerator
| enumerator_list ',' enumerator
;

enumerator
: IDENTIFIER
| IDENTIFIER '=' constant_expression
;

type_qualifier
: CONST
| VOLATILE
;

declarator
: pointer direct_declarator
| direct_declarator
;

direct_declarator
: IDENTIFIER
| '(' declarator ')'
| direct_declarator '[' constant_expression ']'
| direct_declarator '[' ']'
| direct_declarator '(' parameter_type_list ')'
| direct_declarator '(' identifier_list ')'
| direct_declarator '(' ')'
;

pointer
: '*'
| '*' type_qualifier_list
| '*' pointer
| '*' type_qualifier_list pointer
;

type_qualifier_list
: type_qualifier
| type_qualifier_list type_qualifier
;

parameter_type_list
: parameter_list
| parameter_list ',' ELLIPSIS
;

parameter_list
: parameter_declaration
| parameter_list ',' parameter_declaration
;

parameter_declaration
: declaration_specifiers declarator

```

```

| declaration_specifiers abstract_declarator
| declaration_specifiers
;

identifier_list
: IDENTIFIER
| identifier_list ',' IDENTIFIER
;

type_name
: specifier_qualifier_list
| specifier_qualifier_list abstract_declarator
;

abstract_declarator
: pointer
| direct_abstract_declarator
| pointer direct_abstract_declarator
;

direct_abstract_declarator
: '(' abstract_declarator ')'
| '[' ']'
| '[' constant_expression ']'
| direct_abstract_declarator '[' ']'
| direct_abstract_declarator '['
  constant_expression ']'
| '(' ')'
| '(' parameter_type_list ')'
| direct_abstract_declarator '(' ')'
| direct_abstract_declarator '(' parameter_type_list ')'
;

initializer
: assignment_expression
| '{' initializer_list '}'
| '{' initializer_list ',' '}'
;

initializer_list
: initializer
| initializer_list ',' initializer
;

statement
: labeled_statement
| compound_statement
| expression_statement
| selection_statement
| iteration_statement
| jump_statement
;

labeled_statement
: IDENTIFIER ':' statement
| CASE constant_expression ':' statement
| DEFAULT ':' statement
;

compound_statement
: '{' '}'
| '{' statement_list '}'
| '{' declaration_list '}'
| '{' declaration_list statement_list '}'
;

declaration_list
: declaration
| declaration_list declaration
;

statement_list
: statement
| statement_list statement
;

expression_statement
: ';'
| expression ';'
;

selection_statement
: IF '(' expression ')' statement
| IF '(' expression ')' statement ELSE statement
| SWITCH '(' expression ')' statement
;

iteration_statement
: WHILE '(' expression ')' statement
| DO statement WHILE '(' expression ')' ';'
| FOR '(' expression_statement
  expression_statement ')' statement
| FOR '(' expression_statement
  expression_statement expression ')' statement
;

jump_statement
: GOTO IDENTIFIER ';'
| CONTINUE ';'
| BREAK ';'
| RETURN ';'
| RETURN expression ';'
;

translation_unit
: external_declaration
| translation_unit external_declaration
;

external_declaration
: function_definition
| declaration
;

function_definition
: declaration_specifiers declarator declaration_list
  compound_statement
| declaration_specifiers declarator compound_statement
| declarator declaration_list compound_statement
| declarator compound_statement
;

```