C−−

ZoukiRi

February 2, 2021

1 Keywords and Operators of C−−

TO BE CONTINUED

2 Syntax of C−−

```
      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
                         ;
argument_expression_list : assignment_expression
                         | argument_expression_list ‘,’ assignment_expression
                         ;
        unary_expression : postfix_expression
                         | unary_operator cast_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 : specifier ‘;’
                        | specifier init_declarator_list ‘;’
                        ;
   init_declarator_list : init_declarator
                        | init_declarator_list ‘,’ init_declarator
                        ;
```

```
         init_declarator : declarator
                         | declarator ' = ' initializer
                         ;
               specifier : VOID
                         | INT
                         | FLOAT
                         | struct_specifier
                         ;
        struct_specifier
                         : STRUCT IDENTIFIER '{' struct_declaration_list '}'
                         | STRUCT IDENTIFIER
                         ;
 struct_declaration_list : struct_declaration
                         | struct_declaration_list struct_declaration
                         ;
      struct_declaration : specifier struct_declarator_list ';'
                         ;
  struct_declarator_list : struct_declarator
                         | struct_declarator_list ',' declarator
                         ;
              declarator : pointer direct_declarator
                         | direct_declarator
                         ;
       direct_declarator : IDENTIFIER
                         | direct_declarator '(' ')'
                         | direct_declarator '(' parameter_type_list ')'
                         ;
                 pointer : ' * '
                         | ' * ' pointer
                         ;parameter_list                                        :
                         | parameter_list ',' parameter_declaration
                         ;
   parameter_declaration : specifier declarator
                         | specifier abstract_declarator
                         | specifier
                         ;
         identifier_list : IDENTIFIER
```

```
                     | identifier_list ‘,’ IDENTIFIER
                     ;
           type_name : specifier
                     | specifier abstract_declarator
                     ;
 abstract_declarator : pointer
                     ;
         initializer : assignment_expression
                     ;
    initializer_list : initializer
                     | initializer_list ‘,’ initializer
                     ;
           statement : compound_statement
                     | expression_statement
                     | selection_statement
                     | iteration_statement
                     | jump_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
                     ;
 iteration_statement : WHILE ‘(’ expression ‘)’ statement
                     | DO statement WHILE ‘(’ expression ‘)’ ‘;’
                     | FOR ‘(’ expression_statement expression_statement ‘)’ statement
                     | FOR ‘(’ expression_statement expression_statement expression ‘)’ statement
                     ;
      jump_statement : CONTINUE ‘;’
                     | BREAK ‘;’
                     | RETURN ‘;’
                     | RETURN expression ‘;’
                     ;
    translation_unit : external_declaration
                     | translation_unit external_declaration
                     ;
external_declaration : function_definition
                     | declaration
                     ;
 function_definition : specifier declarator compound_statement
                     ;
```