과제 – 5 (C언어 구문분석기)

컴퓨터학부 20160548 이승현

1. 문제 해결 방법

이 과제의 목표는 yacc과 lex 프로그램을 사용하여 C언어 구문 분석기를 만드는 것이다. 이를 위하여 yacc 명세서와 lex 명세서를 만들었다. yacc 명세서는 my_c.y, lex 명세서는 my_c.l이라고 명명했다. 이 두 명세서와 yacc, lex 프로그램을 이용하여 C언어로 된 y.tab.c, y.tab.h, lex.yy.c 파일을 생성하고, 이 파일들을 컴파일하여 최종적으로 C언어 구문 분석기를 생성한다.

- yacc

첫번째로 yacc 명세서인 my_c.y를 작성한다. yacc 명세서의 선언부에서는 에러 내용 출력에 사용할 수 있도록 yyerror() 함수와 extern int line_no 변수를 선언하였다. lex 프로그램을 이용해 생성될 어휘분석기에서 개행 문자를 읽을 때마다 이 line_no를 1씩 증가시킨다. 에러 발생 시에 yyerror() 함수에서 line_no를 출력해 몇 번째 줄 근처에서 에러가 발생했는지 확인할 수 있다. 추가적으로 int yylex (void);를 선언하고 stdio.h를 include하여 나중에 제대로 컴파일이 될 수 있도록 하였다. 또 start statement인 program을 %start 뒤에 적고, 사용되는 모든 terminal symbol들을 %token 뒤에 적었다.

명세서의 문법과 번역규칙부에서는 교재에 있는 C언어 문법의 BNF을 그대로 옮겨 적었다. 다만, 교재의 2.4 초기화 수식 부분에서 "{ initializer }"가 "initializer"로 reduce 된다고 되어있는 것을 "{ initializer_list }"가 "initializer"로 reduce 되도록 수정했다.

명세서의 보조 프로그램부 역시 교재의 내용과 동일하게 main(), yyerror()를 정의하고 extern char *yytext;를 선언하였다.

- lex

다음은 lex 명세서인 my_c.l을 작성한다. lex 명세서 역시 교재의 내용을 그대로 옮겨 적었다. 그 외에 추가로 yywrap() 함수와 checkIdentifier() 함수를 다음과 같이 정의하였다.

```
int checkIdentifier(char *s) {
        char *table[] = {"int", "float", "char", "void"};
        int i;
        int table_size = sizeof(table) / sizeof(table[0]);

        for (i = 0; i < table_size; ++i) {
```

```
                if (!strcmp(table[i], s)) {
                        return (TYPE_IDENTIFIER);
                }
        }
        return (IDENTIFIER);
}

int yywrap() {
        return 1;
}
```

checkIdentifier() 함수는 위에서 볼 수 있듯이 어휘분석기가 문자열을 읽었을 때 type identifier들이 저장된 table을 순회하며 읽어 들인 문자열과 일일이 비교한다. 테이블에 저장된 type identifier와 읽은 문자열이 일치하면 TYPE_IDENTIFIER를 리턴하고, 그렇지 않으면 IDENTIFIER를 리턴 하도록 했다.

마지막으로 "yacc -d my_c.y" 명령어를 이용해 y.tab.c와 y.tab.h를 생성하고, "lex my_c.l" 명령어로 lex.yy.c 파일을 생성했다. 그 다음 최종적으로 "gcc y.tab.c lex.yy.c" 명령어를 이용하여 C언어 구문분석기 프로그램을 생성했다.

2. 실행 결과

- **일반 선언문 테스트**

```
shlee@shlee-virtual-machine:~/workspace/ssucompiler/project5$ cat test1.c
int a, b, c = 0;
float x = 1, y, z = 2;
char j = 'j';
char ch = "hello";
shlee@shlee-virtual-machine:~/workspace/ssucompiler/project5$ ./a.out < test1.c
shlee@shlee-virtual-machine:~/workspace/ssucompiler/project5$ cat test1-1.c
```

아무 문제없이 구문 분석기가 종료된 것을 확인하였다.

- **잘못된 타입 명칭 테스트**

```
shlee@shlee-virtual-machine:~/workspace/ssucompiler/project5$ cat test1-1.c
int a, b, c = 0;
float x = 1, y, z = 2;
char j = 'j';
char ch = "hello";
my_int i = 0;
shlee@shlee-virtual-machine:~/workspace/ssucompiler/project5$ ./a.out < test1-1.c
line 5 syntax error near i
```

잘못된 타입 명칭(my_int)이 쓰인 곳에 에러 메시지가 표시된 것을 확인하였다.

- **switch문 테스트**

```
shlee@shlee-virtual-machine:~/workspace/ssucompiler/project5$ cat test2.c
int main(void) {
        int a = 0;

        switch (a) {
                case 1:
                        a = 0;
                        break;
                case 2:
                        a = 0;
                        break;
                case 3:
                        a = 0;
                        break;
                case 4:
                        a = 0;
                        break;
                default:
                        break;
        }

        return 0;
}
shlee@shlee-virtual-machine:~/workspace/ssucompiler/project5$ ./a.out < test2.c
shlee@shlee-virtual-machine:~/workspace/ssucompiler/project5$ cat test2-1.c
```

아무 문제없이 구문 분석기가 종료된 것을 확인하였다.

- **case 레이블이 잘못 사용된 경우 테스트**

```
shlee@shlee-virtual-machine:~/workspace/ssucompiler/project5$ cat test2-1.c
int main(void) {
        switch (a) {
                case :
                        a = 0;
                        break;
                case 4:
                        a = 0;
                        break;
        }
}
shlee@shlee-virtual-machine:~/workspace/ssucompiler/project5$ ./a.out < test2-1.c
line 3 syntax error near :
```

잘못된 case 레이블이 쓰인 곳에 에러 메시지가 표시된 것을 확인하였다.

- **복합문 내에서 명령문이 선언문보다 먼저 나온 경우 테스트**

```
shlee@shlee-virtual-machine:~/workspace/ssucompiler/project5$ cat test5.c
int main() {
        5 + 2;
        int a;
}
shlee@shlee-virtual-machine:~/workspace/ssucompiler/project5$ ./a.out < test5.c
line 3 syntax error near int
```

잘못된 위치에 에러 메시지가 표시된 것을 확인하였다.

- **if문의 괄호 내에 수식이 없는 경우 테스트**

```
shlee@shlee-virtual-machine:~/workspace/ssucompiler/project5$ cat test6.c
int main() {
        if();
}
shlee@shlee-virtual-machine:~/workspace/ssucompiler/project5$ ./a.out < test6.c
line 2 syntax error near )
```

잘못된 위치에 에러 메시지가 표시된 것을 확인하였다.

- **세미콜론이 없는 경우 테스트**

```
shlee@shlee-virtual-machine:~/workspace/ssucompiler/project5$ cat test7.c
int main() {
        int a = 3 + 2
}
shlee@shlee-virtual-machine:~/workspace/ssucompiler/project5$ ./a.out < test7.c
line 3 syntax error near }
```

잘못된 위치에 에러 메시지가 표시된 것을 확인하였다.

- **정상적으로 작성된 프로그램 1**

test4.c – 문법적으로 문제없는 일반 선언문, 함수 선언문, 선택문(if, if else, switch), 반복문(while, for) 분기문(break, continue, return) 등 테스트

```
int a;
void a;
float a;
int a = 10;
void a = 10 * 2;
float a = 'c';
char *str = "hello world";

void a();
int a(int, char, void);
float a(int a, char b, float c);

int main() {
	int a;
	float a;
	char a;
	int a[];
	int a[]()[];
	int a[3]()[44444];
	float a[1 + 2 * 3][2] = {{1, 2, 3}, {4+2}, 1, 1};

	;;;;;;;;;;;;;;;;;;;;;;;;;;
	,,,,,,,,,,,,,,,,,,,,,,,,,,

	while(1);
	while(1)
		printf("hello world");
	while(1){
```

```
        printf("hello world");
}
while(1)
        break;
while(1)
        continue;

for(;;);
for(1;2;3);
for(i = 1;i < 100;++i);
for(i = 1;i < 100;++i)
        printf("hello world");
for(;;)
        continue;
for(;;)
        break;

if (1);
if (1)
        printf("hello world");
if (1) {
        printf("hello world");
}

if (1);
else;
if (1)
        printf("if");
else
        printf("else");
```

```
        if (1) {
                printf("hello world");
        } else {
                printf("hello world");
        }

        switch(a) {
                case 1:
                        break;
                case 2:
                        break;
                case 3:
                        break;
                default:
                        ;
        }
        return 0;
}
```

위 소스코드를 검사한 결과

```
shlee@shlee-virtual-machine:~/workspace/ssucompiler/project5$ ./a.out < test4.c
shlee@shlee-virtual-machine:~/workspace/ssucompiler/project5$
```

아무 문제없이 구문 분석기가 종료된 것을 확인하였다.

- **정상적으로 작성된 프로그램 2**

test3.c – 운영체제 과목의 과제로 작성했던 간단한 shell 프로그램의 소스코드에서 우리가 만든 컴파일러가 허용하지 않는 문장들만 삭제한 뒤 테스트.

```
void execute_command(char **tokens, int command_start_index, int stdin_fd);
int get_next_pipe_index(char **tokens, int command_start_index);
int check_exit_status(int status);
```

```
char **tokenize(char *line)
{
   char **tokens = (char **)malloc(MAX_NUM_TOKENS * sizeof(char *));
   char *token = (char *)malloc(MAX_TOKEN_SIZE * sizeof(char));
   int i, tokenIndex = 0, tokenNo = 0;

   for(i =0; i < strlen(line); i++){

      char readChar = line[i];

      if (readChar == ' ' || readChar == 'n' || readChar == 't'){
         token[tokenIndex] = '0';
         if (tokenIndex != 0){
            tokens[tokenNo] = (char*)malloc(MAX_TOKEN_SIZE*sizeof(char));
            strcpy(tokens[tokenNo++], token);
            tokenIndex = 0;
         }
      } else {
         token[tokenIndex++] = readChar;
      }
   }

   free(token);
   tokens[tokenNo] = NULL;
   return tokens;
}


int main(int argc, char* argv[]) {
            char   line[MAX_INPUT_SIZE];
```

```
char    **tokens;
int i;
char current_dir_name[BUFFER_SIZE];
char *path_env_value;
char new_path_env_value[BUFFER_SIZE];

getcwd(current_dir_name, BUFFER_SIZE);
path_env_value = getenv("PATH");
sprintf(new_path_env_value, "%s:%s", path_env_value, current_dir_name);
setenv("PATH", new_path_env_value, 1);

FILE* fp;
if(argc == 2) {
        fp = fopen(argv[1],"r");
        if(fp < 0) {
                printf("File doesn't exists.");
                return -1;
        }
}


while(1) {
        bzero(line, sizeof(line));
        if(argc == 2) {
                if(fgets(line, sizeof(line), fp) == NULL) {
                        break;
                }
                line[strlen(line) - 1] = '0';
        } else {
                printf("$ ");
```

```
                scanf("%[^n]", line);
                getchar();
            }

            line[strlen(line)] = 'n';
            tokens = tokenize(line);

            execute_command(tokens, 0, 0);

            for(i=0;tokens[i]!=NULL;i++){
                free(tokens[i]);
            }
            free(tokens);

        }
        return 0;
}

void execute_command(char **tokens, int command_start_index, int stdin_fd){
        int pid;
        int status;
        int pipe_fd[2];
        int pipe_index;

        if (!tokens[command_start_index]) {
            return;
        }

        if ((pipe_index = get_next_pipe_index(tokens, command_start_index)) > 0) {
            tokens[pipe_index] = NULL;
```

```
            if (pipe(pipe_fd) == -1) {
                        fprintf(stderr, "pipe() error.n");
            }
}

if ((pid = fork()) > 0) {
            waitpid(pid, &status, WUNTRACED);
            if (!check_exit_status(status)){
                        return;
            }
            if (pipe_index > 0) {
                        close(pipe_fd[1]);
                        execute_command(tokens, pipe_index + 1, pipe_fd[0]);
                        close(pipe_fd[0]);
            }
} else if (pid == 0) {
            if (pipe_index > 0) {
                        close(pipe_fd[0]);
                        if (dup2(pipe_fd[1], 1) != 1) {
                                    fprintf(stderr, "dup2() error 2n");
                        }
            }

            if (stdin_fd > 0) {
                        dup2(stdin_fd, 0);
            }

            if (execvp(tokens[command_start_index], tokens + command_start_index) < 0) {
                        fprintf(stderr, "SSUShell : Incorrect commandn");
                        exit(1);
```

```
                }

        } else {
                fprintf(stderr, "fork() error.n");
        }

        return;
}

int check_exit_status(int status) {
        if (WIFEXITED(status)) {
                return 1;
        } else if (WIFSIGNALED(status)) {
                fprintf(stderr, "abnormal termination, signal number = %d%sn",
                                WTERMSIG(status), "");
                return 0;
        } else if (WIFSTOPPED(status)) {
                fprintf(stderr, "child stopped, signal number = %dn", WSTOPSIG(status));
                return 0;
        }

        return 0;
}

int get_next_pipe_index(char **tokens, int command_start_index){
        int pipe_index = -1;
        int token_index = command_start_index;

        while (tokens[token_index]) {
                if (!strcmp(tokens[token_index], "|")) {
```

```
                    pipe_index = token_index;
                    break;
                }
                ++token_index;
            }

            return pipe_index;
}
```

위 소스코드를 검사한 결과

```
shlee@shlee-virtual-machine:~/workspace/ssucompiler/project5$ ./a.out < test3.c
shlee@shlee-virtual-machine:~/workspace/ssucompiler/project5$
```

아무 문제없이 구문 분석기가 종료된 것을 확인하였다.

3. 소스코드

my_c.l

```
digit           [0-9]
letter          [a-zA-Z_]
delim           [ ₩t]
line            [₩n]
ws              {delim}+
%{
#include "y.tab.h"
//extern int yylval;
int line_no = 1;
//char *makeString();
int checkIdentifier();
%}
%%
{ws}            { }
{line}          { line_no++;}
auto            { return(AUTO_SYM); }
break           { return(BREAK_SYM); }
case            { return(CASE_SYM); }
continue                { return(CONTINUE_SYM); }
default         { return(DEFAULT_SYM); }
do              { return(DO_SYM); }
else            { return(ELSE_SYM); }
enum            { return(ENUM_SYM); }
for             { return(FOR_SYM); }
if              { return(IF_SYM); }
return          { return(RETURN_SYM); }
```

```
sizeof          { return(SIZEOF_SYM); }
static          { return(STATIC_SYM); }
struct          { return(STRUCT_SYM); }
switch          { return(SWITCH_SYM); }
typedef         { return(TYPEDEF_SYM); }
union           { return(UNION_SYM); }
while           { return(WHILE_SYM); }
"₩+₩+"                  { return(PLUSPLUS); }
"₩-₩-"          { return(MINUSMINUS); }
"₩->"           { return(ARROW); }
"<"             { return(LSS); }
">"             { return(GTR); }
"<="            { return(LEQ); }
">="            { return(GEQ); }
"=="            { return(EQL); }
"!="            { return(NEQ); }
"&&"            { return(AMPAMP); }
"||"            { return(BARBAR); }
"₩.₩.₩."                { return(DOTDOTDOT); }
"₩("            { return(LP); }
"₩)"            { return(RP); }
"₩["            { return(LB); }
"₩]"            { return(RB); }
"₩{"            { return(LR); }
"₩}"            { return(RR); }
"₩:"            { return(COLON); }
"₩."            { return(PERIOD); }
"₩,"            { return(COMMA); }
```

```
"₩!"                    { return(EXCL); }
"₩*"                    { return(STAR); }
"₩/"                    { return(SLASH); }
"₩%"                    { return(PERCENT); }
"₩&"                    { return(AMP); }
"₩;"                    { return(SEMICOLON); }
"₩+"                    { return(PLUS); }
"₩-"                                { return(MINUS); }
"₩="                                { return(ASSIGN); }
{digit}+                        { return(INTEGER_CONSTANT);}
{digit}+₩.{digit}+                          { return(FLOAT_CONSTANT);}
{letter}({letter}|{digit})*     { return(checkIdentifier(yytext));}
₩"([^"₩n]|₩₩["₩n])*₩"                       { return(STRING_LITERAL);}
₩'([^'₩n]|₩'₩')₩'                           { return(CHARACTER_CONSTANT);}
%%
int checkIdentifier(char *s) {
        char *table[] = {"int", "float", "char", "void"};
        int i;
        int table_size = sizeof(table) / sizeof(table[0]);


        for (i = 0; i < table_size; ++i) {
                if (!strcmp(table[i], s)) {
                        return (TYPE_IDENTIFIER);
                }
        }


        return (IDENTIFIER);
}
```

```
int yywrap() {
        return 1;
}
```

my_c.y

```
%{
#include "y.tab.h"
#include <stdio.h>
extern int line_no;
extern int yylex (void);
void yyerror(char *s);
%}
%start program
%token AUTO_SYM BREAK_SYM CASE_SYM CONTINUE_SYM DEFAULT_SYM DO_SYM ELSE_SYM ENUM_SYM FOR_SYM
IF_SYM RETURN_SYM SIZEOF_SYM STATIC_SYM STRUCT_SYM SWITCH_SYM TYPEDEF_SYM UNION_SYM WHILE_SYM
PLUSPLUS MINUSMINUS ARROW LSS GTR LEQ GEQ EQL NEQ AMPAMP BARBAR DOTDOTDOT LP RP LB RB LR RR COLON
PERIOD COMMA EXCL STAR SLASH PERCENT AMP SEMICOLON PLUS MINUS ASSIGN INTEGER_CONSTANT
FLOAT_CONSTANT STRING_LITERAL CHARACTER_CONSTANT IDENTIFIER TYPE_IDENTIFIER
%%
program
        : translation_unit
translation_unit
        : external_declaration
        | translation_unit external_declaration
external_declaration
        : function_definition
        | declaration

function_definition
        : declaration_specifiers declarator compound_statement
        | declarator compound_statement

declaration
        : declaration_specifiers SEMICOLON
```

```
        | declaration_specifiers init_declarator_list SEMICOLON
declaration_specifiers
        : type_specifier
        | storage_class_specifier
        | type_specifier declaration_specifiers
        | storage_class_specifier declaration_specifiers
storage_class_specifier
        : AUTO_SYM
        | STATIC_SYM
        | TYPEDEF_SYM
init_declarator_list
        : init_declarator
        | init_declarator_list COMMA init_declarator
init_declarator
        : declarator
        | declarator ASSIGN initializer

type_specifier
        : struct_specifier
        | enum_specifier
        | TYPE_IDENTIFIER

struct_specifier
        : struct_or_union IDENTIFIER LR struct_declaration_list RR
        | struct_or_union LR struct_declaration_list RR
        | struct_or_union IDENTIFIER
struct_or_union
        : STRUCT_SYM
```

```
        | UNION_SYM
struct_declaration_list
        : struct_declaration
        | struct_declaration_list struct_declaration
struct_declaration
        : type_specifier struct_declarator_list SEMICOLON
struct_declarator_list
        : struct_declarator
        | struct_declarator_list COMMA struct_declarator
struct_declarator
        : declarator

enum_specifier
        : ENUM_SYM IDENTIFIER LR enumerator_list RR
        | ENUM_SYM LR enumerator_list RR
        | ENUM_SYM IDENTIFIER
enumerator_list
        : enumerator
        | enumerator_list COMMA enumerator
enumerator
        : IDENTIFIER
        | IDENTIFIER ASSIGN constant_expression

declarator
        : pointer direct_declarator
        | direct_declarator
pointer
        : STAR
```

| STAR pointer

direct_declarator

: IDENTIFIER

| LP declarator RP

| direct_declarator LB constant_expression_opt RB

| direct_declarator LP parameter_type_list_opt RP

constant_expression_opt

:

| constant_expression

parameter_type_list_opt

:

| parameter_type_list

parameter_type_list

: parameter_list

| parameter_list DOTDOTDOT

parameter_list

: parameter_declaration

| parameter_list COMMA parameter_declaration

parameter_declaration

: declaration_specifiers declarator

| declaration_specifiers abstract_declarator

| declaration_specifiers

abstract_declarator

: pointer

| direct_abstract_declarator

| pointer direct_abstract_declarator

direct_abstract_declarator

: LP abstract_declarator RP

| LB constant_expression_opt RB

| LP parameter_type_list_opt RP

| direct_abstract_declarator LB constant_expression_opt RB

| direct_abstract_declarator LP parameter_type_list_opt RP

initializer

: constant_expression

| LR initializer_list RR

initializer_list

: initializer

| initializer_list COMMA initializer

statement

: labeled_statement

| compound_statement

| expression_statement

| selection_statement

| iteration_statement

| jump_statement

labeled_statement

: CASE_SYM constant_expression COLON statement

| DEFAULT_SYM COLON statement

compound_statement

: LR declaration_list statement_list RR

declaration_list

:

| declaration_list declaration

statement_list

:

| statement_list statement

expression_statement

: SEMICOLON

| expression SEMICOLON

selection_statement

: IF_SYM LP expression RP statement

| IF_SYM LP expression RP statement ELSE_SYM statement

| SWITCH_SYM LP expression RP statement

iteration_statement

: WHILE_SYM LP expression RP statement

| DO_SYM statement WHILE_SYM LP expression RP SEMICOLON

| FOR_SYM LP expression_opt SEMICOLON expression_opt SEMICOLON expression_opt RP statement

expression_opt

:

| expression

jump_statement

: RETURN_SYM expression_opt SEMICOLON

| CONTINUE_SYM SEMICOLON

| BREAK_SYM SEMICOLON

primary_expression

: IDENTIFIER

| INTEGER_CONSTANT

| FLOAT_CONSTANT

| CHARACTER_CONSTANT

| STRING_LITERAL

| LP expression RP

postfix_expression

: primary_expression

| postfix_expression LB expression RB

| postfix_expression LP arg_expression_list_opt RP

| postfix_expression PERIOD IDENTIFIER

| postfix_expression ARROW IDENTIFIER

| postfix_expression PLUSPLUS

| postfix_expression MINUSMINUS

arg_expression_list_opt

:

| arg_expression_list

arg_expression_list

: assignment_expression

| arg_expression_list COMMA assignment_expression

unary_expression

: postfix_expression

| PLUSPLUS unary_expression

| MINUSMINUS unary_expression

| AMP cast_expression

```
        | STAR cast_expression
        | EXCL cast_expression
        | MINUS cast_expression
        | PLUS cast_expression
        | SIZEOF_SYM unary_expression
        | SIZEOF_SYM LP type_name RP
cast_expression
        : unary_expression
        | LP type_name RP cast_expression
type_name
        : declaration_specifiers
        | declaration_specifiers abstract_declarator

multiplicative_expression
        : cast_expression
        | multiplicative_expression STAR cast_expression
        | multiplicative_expression SLASH cast_expression
        | multiplicative_expression PERCENT cast_expression
additive_expression
        : multiplicative_expression
        | additive_expression PLUS multiplicative_expression
        | additive_expression MINUS multiplicative_expression

relational_expression
        : additive_expression
        | relational_expression LSS additive_expression
        | relational_expression GTR additive_expression
        | relational_expression LEQ additive_expression
```

```
        | relational_expression GEQ additive_expression
equality_expression
        : relational_expression
        | equality_expression EQL relational_expression
        | equality_expression NEQ relational_expression

logical_and_expression
        : equality_expression
        | logical_and_expression AMPAMP equality_expression
logical_or_expression
        : logical_and_expression
        | logical_or_expression BARBAR logical_and_expression

constant_expression
        : expression
expression
        : assignment_expression
assignment_expression
        : logical_or_expression
        | unary_expression ASSIGN expression

%%
extern char *yytext;
void yyerror(char *s) { printf("line %d %s near %s ₩n", line_no, s, yytext); }
int main() {
  yyparse();
}
```

y.tab.h

```
#define AUTO_SYM 257
#define BREAK_SYM 258
#define CASE_SYM 259
#define CONTINUE_SYM 260
#define DEFAULT_SYM 261
#define DO_SYM 262
#define ELSE_SYM 263
#define ENUM_SYM 264
#define FOR_SYM 265
#define IF_SYM 266
#define RETURN_SYM 267
#define SIZEOF_SYM 268
#define STATIC_SYM 269
#define STRUCT_SYM 270
#define SWITCH_SYM 271
#define TYPEDEF_SYM 272
#define UNION_SYM 273
#define WHILE_SYM 274
#define PLUSPLUS 275
#define MINUSMINUS 276
#define ARROW 277
#define LSS 278
#define GTR 279
#define LEQ 280
#define GEQ 281
#define EQL 282
#define NEQ 283
```

```
#define AMPAMP 284
#define BARBAR 285
#define DOTDOTDOT 286
#define LP 287
#define RP 288
#define LB 289
#define RB 290
#define LR 291
#define RR 292
#define COLON 293
#define PERIOD 294
#define COMMA 295
#define EXCL 296
#define STAR 297
#define SLASH 298
#define PERCENT 299
#define AMP 300
#define SEMICOLON 301
#define PLUS 302
#define MINUS 303
#define ASSIGN 304
#define INTEGER_CONSTANT 305
#define FLOAT_CONSTANT 306
#define STRING_LITERAL 307
#define CHARACTER_CONSTANT 308
#define IDENTIFIER 309
#define TYPE_IDENTIFIER 310
```