

# 컴파일러설계 프로젝트1 (Scanner) 보고서

2019019016 서시언

## 1. Compilation method and environment

- Ubuntu 20.04.3 LTS

## 2. Explanation about how to implement and how to operate

- 제공된 Makefile 사용

### [Method 1 (C code)]

- main.c

- NO\_PARSE, TraceScan을 TRUE로 설정한다.

```
10 /* set NO_PARSE to TRUE to get a scanner-only compiler */
11 #define NO_PARSE TRUE
12 /* set NO_ANALYZE to TRUE to get a parser-only compiler */
13 #define NO_ANALYZE FALSE
14
15 /* set NO_CODE to TRUE to get a compiler that does not
16 * generate code
17 */
18 #define NO_CODE FALSE
19
20 #include "util.h"
21 #if NO_PARSE
22 #include "scan.h"
23 #else
24 #include "parse.h"
25 #if !NO_ANALYZE
26 #include "analyze.h"
27 #if !NO_CODE
28 #include "cgen.h"
29 #endif
30 #endif
31 #endif
32
33 /* allocate global variables */
34 int lineno = 0;
35 FILE * source;
36 FILE * listing;
37 FILE * code;
38
39 /* allocate and set tracing flags */
40 int EchoSource = FALSE;
41 int TraceScan = TRUE;
42 int TraceParse = FALSE;
43 int TraceAnalyze = FALSE;
44 int TraceCode = FALSE;
45
```

- 예시 출력(result.1.txt, result.2.txt)에 맞게 내용을 수정해준다.

```
62 }
63 listing = stdout; /* send listing to screen */
64 fprintf(listing, "\nC-MINUS COMPILATION: %s\n", pgm);
```

- global.h

- Tiny's Token(THEN, REPEAT, UNTIL, WRITE, READ, END)을 지우고, C-Minus token을 추가해주고, MAXRESERVED를 6으로 설정한다.

```
25 /* MAXRESERVED - the number of reserved words */
26 #define MAXRESERVED 6
27
28 typedef enum
29     /* book-keeping tokens */
30     {ENDFILE, ERROR,
31      /* reserved words */
32      IF, ELSE, WHILE, RETURN, INT, VOID,
33      /* multicharacter tokens */
34      TD_NUM,
35      /* special symbols */
36      ASSIGN, EQ, NE, LT, LE, GT, GE, PLUS, MINUS, TIMES, OVER, LPAREN, RPAREN, LBRACE, RBRACE, LCURLY, RCURLY, SEMI, COMMA
37      } TokenType;
```

- util.c

- 변경한 토큰들을 반영해 printToken()을 수정한다.

- sepecial symbol은 아래와 같다.

```
ASSIGN, EQ, NE, LT, LE, GT, GE, PLUS, MINUS, TIMES, OVER, LPAREN, RPAREN, LBRACE, RBRACE, LCURLY, RCURLY, SEMI, COMMA
= == != < <= > >= + - * / ( ) [ ] { }
```

```

15 void printToken( TokenType token, const char* tokenString )
16 { switch (token)
17 { case IF:
18 case ELSE:
19 case WHILE:
20 case RETURN:
21 case INT:
22 case VOID:
23     fprintf(listing,
24         "reserved word: %s\n",tokenString);
25     break;
26 case ASSIGN: fprintf(listing,="\\n"); break;
27 case LT: fprintf(listing,"<\\n"); break;
28 case NE: fprintf(listing,"!=\\n"); break;
29 case EQ: fprintf(listing,"==\\n"); break;
30 case LE: fprintf(listing,"<=\\n"); break;
31 case GT: fprintf(listing,">\\n"); break;
32 case GE: fprintf(listing,">=\\n"); break;
33 case LPAREN: fprintf(listing,"(\\n"); break;
34 case RPAREN: fprintf(listing,")\\n"); break;
35 case LBRACE: fprintf(listing,"[\\n"); break;
36 case RBRACE: fprintf(listing,"]\\n"); break;
37 case LCURLY: fprintf(listing,"{\\n"); break;
38 case RCURLY: fprintf(listing,"}\\n"); break;
39 case SEMI: fprintf(listing,";\\n"); break;
40 case COMMA: fprintf(listing,",\\n"); break;
41 case PLUS: fprintf(listing, "+\\n"); break;
42 case MINUS: fprintf(listing, "-\\n"); break;
43 case TIMES: fprintf(listing, "*\\n"); break;
44 case OVER: fprintf(listing, "/\\n"); break;
45 case ENDFILE: fprintf(listing, EOF\\n ); break;
46 case NUM:
47     fprintf(listing,
48         "NUM, val= %s\\n",tokenString);
49     break;
50 case ID:
51     fprintf(listing,
52         "ID, name= %s\\n",tokenString);
53     break;
54 case ERROR:
55     fprintf(listing,
56         "ERROR: %s\\n",tokenString);

```

- Scan.c

- state를 추가해준다.

```

12 /* states in scanner DFA */
13 typedef enum
14 { // add INEQ, INNE, INLT, INGT, INOVER, INCOMMENT_
15 // delete INEQUAL
16 { START,INEQ,INNE,INLT,INGT,INOVER,INCOMMENT_,INNUM,INID,DONE }
17 StateType;
18

```

- reserved word를 수정한다.

```

56 /* lookup table of reserved words */
57 static struct
58 { char* str;
59   TokenType tok;
60   reservedWords[MAXRESERVED]
61 = [{"if",IF}, {"else",ELSE}, {"while",WHILE},
62 {"return",RETURN}, {"int",INT}, {"void",VOID},
63 ]
64 // delete THEN, END, REPEAT, UNTIL, READ, WRITE
65 // add WHILE, RETURN, INT, VOID
66

```

- getToken()을 수정해준다.

```

83 TokenType getToken(void)
84 { /* index for storing into tokenString */
85   int tokenStringIndex = 0;
86   /* holds current token to be returned */
87   TokenType currentToken;
88   /* current state - always begins at START */
89   StateType state = START;
90   /* flag to indicate save to tokenString */
91   int save;
92   while (state != DONE)
93   { int c = getNextChar();
94     save = TRUE;
95     switch (state)
96     { case START:
97       if (isdigit(c))
98         state = INNUM;
99       else if (isalpha(c))
100         state = INT;
101       else if (c == '=')
102         // to check if = or ==
103         state = INEQ;
104       else if (c == '!')
105         // to check if ! or !=
106         state = INNE;
107       else if (c == '<')
108         // to check if < or ==
109         state = INLT;
110       else if (c == '>')
111         // to check if > or ==
112         state = INGT;
113       else if (c == '/')
114       {
115         // Since it can be COMMENT(*), don't save
116         save = FALSE;
117         // to check if / or /*
118         state = INOVER;
119       }

```

```

120     else if ((c == ' ') || (c == '\t') || (c == '\n'))
121         save = FALSE;
122     else
123     { state = DONE;
124     switch (c)
125     { case EOF:
126         save = FALSE;
127         currentToken = ENDFILE;
128         break;
129     case '+':
130         currentToken = PLUS;
131         break;
132     case '-':
133         currentToken = MINUS;
134         break;
135     case '*':
136         currentToken = TIMES;
137         break;
138     case '(':
139         currentToken = LPAREN;
140         break;
141     case ')':
142         currentToken = RPAREN;
143         break;
144     case '[':
145         currentToken = LBRACE;
146         break;
147     case ']':
148         currentToken = RBRACE;
149         break;
150     case '{':
151         currentToken = LCURLY;
152         break;
153     case '}':
154         currentToken = RCURLY;
155         break;
156     case ';':
157         currentToken = SEMI;
158         break;
159     case ',':
160         currentToken = COMMA;
161         break;
162     default:
163         currentToken = ERROR;
164         break;
165     }
166 }
167 break;
168 case INOVER:
169     if (c == '*')
170     { // COMMENT /*)
171         save = FALSE;
172         state = INCOMMENT;
173     }
174     else
175     { // OVER */
176         ungetNextChar();
177         state = DONE;
178         currentToken = OVER;
179     }
180     break;
181 case INCOMMENT:
182     save = FALSE;
183     if (c == EOF)
184     { state = DONE;
185         currentToken = ENDFILE;
186     }
187     // to check COMMENT is end
188     if (c == '*') state = INCOMMENT_;
189     break;
190 case INCOMMENT_:
191     save = FALSE;
192     if (c == EOF)
193     { state = DONE;
194         currentToken = ENDFILE;
195     }
196     // end of COMMENT */
197     else if (c == '/') state = START;
198     // still in COMMENT
199     else state = INCOMMENT;
200     break;

```

- 주석일 때는 save를 FALSE로 설정한다. (state가 INCOMMENT, INCOMMENT\_ 일 때 / state가 INOVER 이면서 c가 \*일 때)

```

201     case INEQ:
202         state = DONE;
203         // EQ (==)
204         if (c == '=') currentToken = EQ;
205         else
206         { // ASSIGN (=)
207             ungetNextChar();
208             currentToken = ASSIGN;
209         }
210         break;
211     case INNE:
212         state = DONE;
213         // NE (!=)
214         if (c == '=') currentToken = NE;
215         else
216         { // ERROR !=
217             ungetNextChar();
218             save = FALSE;
219             currentToken = ERROR;
220         }
221         break;

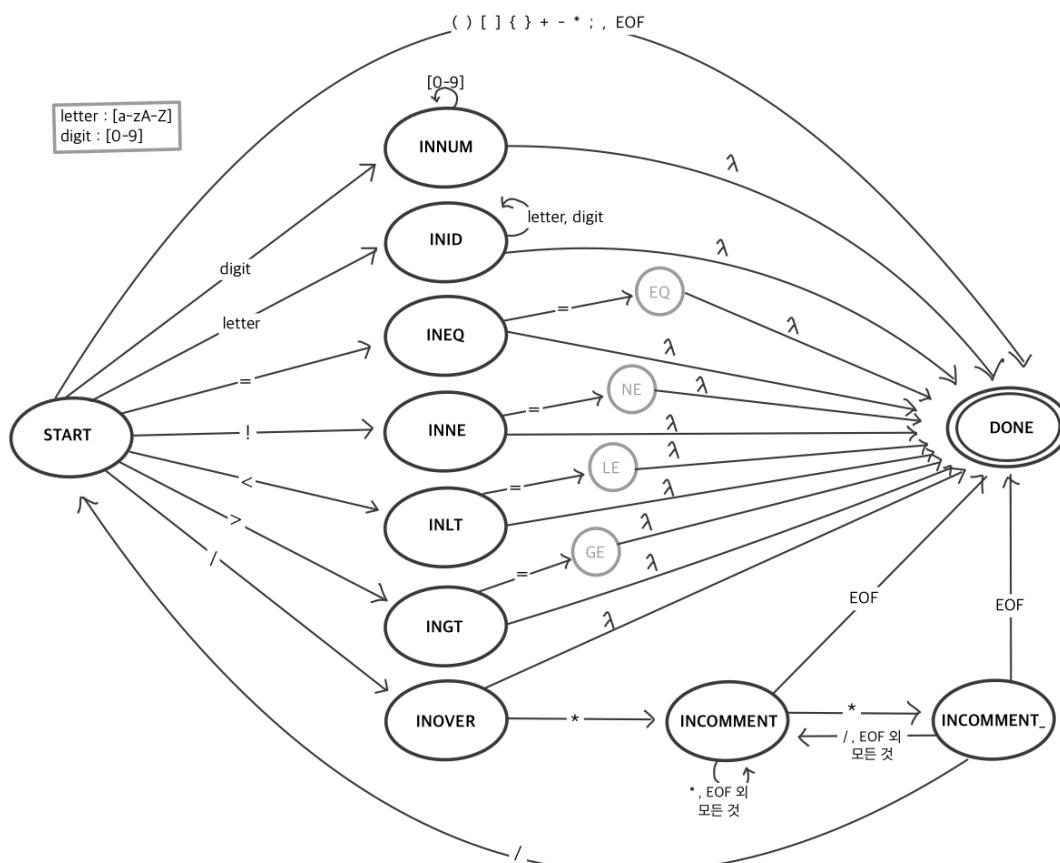
```

```

222
223     case INLT:
224         state = DONE;
225         // LE (<=)
226         if (c == '=') currentToken = LE;
227         else
228             { // LT (<)
229                 ungetNextChar();
230                 currentToken = LT;
231             }
232             break;
233     case INGT:
234         state = DONE;
235         // GE (>=)
236         if (c == '=') currentToken = GE;
237         else
238             { // GT (>)
239                 ungetNextChar();
240                 currentToken = GT;
241             }
242             break;
243     case INNUM:
244         if (!isdigit(c))
245             { /* backup in the input */
246                 ungetNextChar();
247                 save = FALSE;
248                 state = DONE;
249                 currentToken = NUM;
250             }
251             break;
252     case INID:
253         if (!isalpha(c) && !isdigit(c))
254             { /* backup in the input */
255                 ungetNextChar();
256                 save = FALSE;
257                 state = DONE;
258                 currentToken = ID;
259             }
260             break;
261     case DONE:
262         default: /* should never happen */
263             fprintf(listing,"Scanner Bug: state= %d\n",state);
264             state = DONE;
265             currentToken = ERROR;
266             break;
}

```

- getToken()이 나타내는 DFA는 다음과 같다.



- EQ, NE, LE, GE는 State가 아닌 currentToken이 EQ, NE, LE, GE임을 의미함.

- 실행

**make**

**./cminus\_cimpl [input\_file]**

## [Method 2 (Lex(flex))]

- globals.h, main.c, util.c는 Method 1과 동일

- requirement

```
sudo apt-get install flex
```

- cminus.l

- tiny.l을 베이스로 함.

- identifier 수정

```
8
9 digit [0-9]
10 number {digit}+
11 letter [a-zA-Z]
12 /* Identifier start with letter, and then letter or digit */
13 identifier {letter}({letter}|{digit})*
14 newline \
15 whitespace [ \t]
```

- Token rule 수정

```
19 "if" {return IF;}
20 "else" {return ELSE;}
21 "while" {return WHILE;}
22 "return" {return RETURN;}
23 "int" {return INT;}
24 "void" {return VOID;}
25 "=" {return ASSIGN;}
26 "==" {return EQ;}
27 "!=" {return NE;}
28 "<" {return LT;}
29 "<=" {return LE;}
30 ">" {return GT;}
31 ">=" {return GE;}
32 "+" {return PLUS;}
33 "-" {return MINUS;}
34 "*" {return TIMES;}
35 "/" {return OVER;}
36 "(" {return LPAREN;}
37 ")" {return RPAREN;}
38 "[" {return LBRACE;}
39 "]" {return RBRACE;}
40 "{" {return LCURLY;}
41 "}" {return RCURLY;}
42 ";" {return SEMI;}
43 "," {return COMMA;}
44 {number} {return NUM;}
45 {identifier} {return ID;}
46 {newline} {lineno++;}
47 {whitespace} /* skip whitespace */
48 /* { char c;
49 // in COMMENT
50 do
51 { c = input();
52 if (c == '#'){
53 /* to check if end of COMMENT */
54 c = input();
55 /* if end of COMMENT, break */
56 if(c=='\n') break;
57 }
58 /* to cover the case that Lex detect '\0' instead of EOF(-1) in the end of the input source. */
59 if (c == EOF || c == '\0') break;
60 if (c == '\n') lineno++;
61 } while (1);
62 }
```

- 주석을 처리하기 위해 /\* 일 때 EOF가 나오거나, 주석의 끝 \*/이 나올 때까지 while문을 돈다.

- while문 안에서 \*이 나오면 주석의 끝인지 알기 위해 input을 하나 더 확인해본다.

- 이때 \*/이면 주석의 끝이므로 break를 해준다.

- 실행

```
make
```

```
/cminus_lexer [input_file]
```

## 3. Example and Result Screenshot

- test.1.txt

```
1 /* A program to perform Euclid's
2     Algorithm to computer gcd */
3
4 int gcd (int u, int v)
5 {
6     if (v == 0) return u;
7     else return gcd(v,u-u/v*v);
8     /* u-u/v*v == u mod v */
9 }
10
11 void main(void)
12 {
13     int x; int y;
14     x = input(); y = input();
15     output(gcd(x,y));
16 }
```

## [Method 1 (C code)]

```
make
```

```
./cminus_cimpl test.1.txt
```

```
stun@siun-QEMU-Virtual-Machine:~/2021_ele4029_2019019016/1_Scanner$ ./cminus_cimpl test.1.txt
C-MINUS COMPILATION: test.1.txt
 4: reserved word: int
 4: ID, name= gcd
 4: (
 4: reserved word: int
 4: ID, name= u
 4: ,
 4: reserved word: int
 4: ID, name= v
 4: )
 5: {
 6: reserved word: if
 6: (
 6: ID, name= v
 6: ==
 6: NUM, val= 0
 6: )
 6: reserved word: return
 6: ID, name= u
 6: ;
 7: reserved word: else
 7: reserved word: return
 7: ID, name= gcd
 7: (
 7: ID, name= v
 7: ,
 7: ID, name= u
 7: ;
 7: ID, name= u
 7: /
 7: ID, name= v
 7: *
 7: ID, name= v
 7: )
 7: ;
 9: }
11: reserved word: void
11: ID, name= main
11: (
11: reserved word: void
11: )
12: {
13: reserved word: int
13: ID, name= x
13: ;
13: reserved word: int
13: ID, name= y
13: ;
14: ID, name= x
14: =
14: ID, name= input
14: (
14: )
14: ;
14: ID, name= y
14: =
14: ID, name= input
14: (
14: )
14: ;
15: ID, name= output
15: (
15: ID, name= gcd
15: (
15: ID, name= x
15: ,
15: ID, name= y
15: )
15: )
15: ;
16: }
17: EOF
```

█

## [Method 2 (Lex(Flex))]

```
make
```

```
./cminus_lex test.1.txt
```

```
stun@siun-QEMU-Virtual-Machine:~/2021_ele4029_2019019016/1_Scanner$ ./cminus_lex test.1.txt
C-MINUS COMPILATION: test.1.txt
 4: reserved word: int
 4: ID, name= gcd
 4: (
 4: reserved word: int
 4: ID, name= u
 4: ,
 4: reserved word: int
 4: ID, name= v
 4: )
 5: {
 6: reserved word: if
 6: (
 6: ID, name= v
 6: ==
 6: NUM, val= 0
 6: )
 6: reserved word: return
 6: ID, name= u
 6: ;
 7: reserved word: else
 7: reserved word: return
```

```
7: ID, name= gcd
7: (
7: ID, name= v
7: ,
7: ID, name= u
7: -
7: ID, name= u
7: /
7: ID, name= v
7: *
7: ID, name= v
7: )
7: ;
9: }
11: reserved word: void
11: ID, name= main
11: (
11: reserved word: void
11: )
12: {
13: reserved word: int
13: ID, name= x
13: ;
13: reserved word: int
13: ID, name= y
13: ;
14: ID, name= x
14: =
14: ID, name= input
14: (
14: )
14: ;
14: ID, name= y
14: =
14: ID, name= input
14: (
14: )
14: ;
14: ;
15: ID, name= output
15: (
15: ID, name= gcd
15: (
15: ID, name= x
15: ,
15: ID, name= y
15: )
15: )
15: ;
16: }
17: EOF
```