

REPORT

“ CH04 Scanner 구현 ”



과 목 명	컴파일러 설계 및 구축 (월25,26)
담당교수	이양선 교수님
학 과	컴퓨터공학과
학 번	2015305084
이 름	홍송희
제 출 일	2018.10.25

4.11 본문에 있는 함수 scanner()를 이용하여 실질적인 Mini C의 어휘 분석기를 구현하시오. 이때, 입력 프로그램은 부록 A에 있는 prime.mc를 사용하고 출력의 형태는 다음과 같은 형태로 하시오.

<소스 코드>

```
1025
main.c x
1025 (전역 범위) main(int argc, char * argv[])
1 #include <stdio.h>
2 #include <string.h>
3 #include <stdlib.h>
4 #include <ctype.h>
5
6 #define NO_KEYWORDS 7
7 #define ID_LENGTH 12
8
9 void lexicalError(int n);
10 int superLetter(char ch);
11 int superLetterOrDigit(char ch);
12 int getIntNum(char firstCharacter, FILE* source_file);
13
14 char id[ID_LENGTH];
15 char ch;
16
17 struct tokenType {
18     int number;
19     union {
20         char id[ID_LENGTH];
21         int num;
22     } value;
23 };
24
25 char *keyword[NO_KEYWORDS] = { "const", "else", "if", "int", "return", "void", "while" };
26
27 enum tsymbol {
28     tnull = -1,
29     tnot, tnoteq, tmod, tmodAssign, tident, tnumber,
30     tand, tparen, trparen, tmul, tmulAssign, tplus,
31     tinc, taddAssign, tcomma, tminus, tdec, tsubAssign,
32     tdiv, tdivAssign, tsemicolon, tless, tlesse, tassign,
33     tequal, tgreat, tcreate, tbracket, trbracket, teof,
34     tconst, telse, tif, tint, treturn, tvoid,
35     twhile, tbrace, tor, trbrace
36 };
37
38 enum tsymbol tnum[NO_KEYWORDS] = { tconst, telse, tif, tint, treturn, tvoid, twhile };
39
40
41 struct tokenType scanner(FILE* source_file)
42 {
43     struct tokenType token;
44     int i, index;
45
46     token.number = tnull;
47
48     do {
49         while (isspace(ch = fgetc(source_file)));
50         if (superLetter(ch)) {
51             i = 0;
52             do {
53                 if (i < ID_LENGTH) id[i++] = ch;
54                 ch = fgetc(source_file);
55             } while (superLetterOrDigit(ch));
56             if (i >= ID_LENGTH) lexicalError(1);
57             id[i] = '\0';
58         }
```

```
main.c x
1025 (전역 범위) scanner(F
49 do {
50     while (isspace(ch = fgetc(source_file)));
51     if (superLetter(ch)) {
52         i = 0;
53         do {
54             if (i < ID_LENGTH) id[i++] = ch;
55             ch = fgetc(source_file);
56         } while (superLetterOrDigit(ch));
57         if (i >= ID_LENGTH) lexicalError(1);
58         id[i] = '\0';
59         ungetc(ch, stdin);
60         for (index = 0; index < NO_KEYWORDS; index++)
61             if (!strcmp(id, keyword[index])) break;
62         if (index < NO_KEYWORDS)
63             token.number = tnum[index];
64         else {
65             token.number = tidet;
66             strcpy_s(token.value.id, ID_LENGTH, id);
67         }
68     }
69     else if (isdigit(ch)) {
70         token.number = tnumber;
71         token.value.num = getIntNum(ch, source_file);
72     }
73     else {
74         switch (ch) {
75             case '/':
76                 id[0] = ch;
77                 ch = fgetc(source_file);
78                 if (ch == '*')
79                     do {
80                         while (ch != '*') ch = fgetc(source_file);
81                         ch = fgetc(source_file);
82                     } while (ch != '/');
83                 else if (ch == '/')
84                     while (fgetc(source_file) != '\n');
85                 else if (ch == '=') token.number = tdivAssign;
86                 else {
87                     token.number = tdiv;
88                     ungetc(ch, stdin);
89                 }
90                 break;
91             case '!':
92                 id[0] = ch;
93                 ch = fgetc(source_file);
94                 if (ch == '=') { token.number = tnoteq; id[1] = ch; }
95                 else {
96                     token.number = tnot;
97                     ungetc(ch, stdin);
98                 }
99                 break;
100             case 'X':
101                 id[0] = ch;
102                 ch = fgetc(source_file);
103                 if (ch == '=') {
104                     token.number = tmodAssign;
105                     id[1] = ch;
106                 }
107             }
```

```
main.c + x
1025 (전역 범위) sca
100 case 'x':
101     id[0] = ch;
102     ch = fgetc(source_file);
103     if (ch == '=') {
104         token.number = tmodAssign;
105         id[1] = ch;
106     }
107     else {
108         token.number = tmod;
109         ungetc(ch, stdin);
110     }
111     break;
112 case '&':
113     id[0] = ch;
114     ch = fgetc(source_file);
115     if (ch == '&') { token.number = tand; id[1] = ch; }
116     else {
117         lexicalError(2);
118         ungetc(ch, stdin);
119     }
120     break;
121 case '+':
122     id[0] = ch;
123     ch = fgetc(source_file);
124     if (ch == '=') { token.number = tmulAssign; id[1] = ch; }
125     else {
126         token.number = tmul;
127         ungetc(ch, stdin);
128     }
129     break;
130 case '+':
131     id[0] = ch;
132     ch = fgetc(source_file);
133     if (ch == '+') { token.number = tinc; id[1] = ch; }
134     else if (ch == '=') { token.number = taddAssign; id[1] = ch; }
135     else {
136         token.number = tplus;
137         ungetc(ch, stdin);
138     }
139     break;
140 case '-':
141     id[0] = ch;
142     ch = fgetc(source_file);
143     if (ch == '-') { token.number = tdec; id[1] = ch; }
144     else if (ch == '-') { token.number = tsubAssign; id[1] = ch; }
145     else {
146         token.number = tminus;
147         ungetc(ch, stdin);
148     }
149     break;
150 case '<':
151     id[0] = ch;
152     ch = fgetc(source_file);
153     if (ch == '=') { token.number = tlesse; id[1] = ch; }
154     else {
155         token.number = tless;
156         ungetc(ch, stdin);
157     }
158     break;
```

```
main.c -> X
1025 (전역 범위) scanner(FILE *)

159 case '=':
160     id[0] = ch;
161     ch = fgetc(source_file);
162     if (ch == '=') { token.number = tequal; id[1] = ch; }
163     else {
164         token.number = tassign;
165         ungetc(ch, stdin);
166     }
167     break;
168 case '>':
169     id[0] = ch;
170     ch = fgetc(source_file);
171     if (ch == '>') { token.number = tcreate; id[1] = ch; }
172     else {
173         token.number = tcreat;
174         ungetc(ch, stdin);
175     }
176     break;
177 case '!':
178     id[0] = ch;
179     ch = fgetc(source_file);
180     if (ch == '!') { token.number = tor; id[1] = ch; }
181     else {
182         lexicalError(3);
183         ungetc(ch, stdin);
184     }
185     break;
186 case '(': id[0] = ch; token.number = tlparen; break;
187 case ')': id[0] = ch; token.number = trparen; break;
188 case ',': id[0] = ch; token.number = tcomma; break;
189 case ';': id[0] = ch; token.number = tsemicolon; break;
190 case '[': id[0] = ch; token.number = tlbracket; break;
191 case ']': id[0] = ch; token.number = trbracket; break;
192 case '{': id[0] = ch; token.number = tlbrace; break;
193 case '}': id[0] = ch; token.number = trbrace; break;
194 case EOF: token.number = teof; break;
195 default: {
196     printf("Current character : %c", ch);
197     lexicalError(4);
198     break;
199 }
200 }
201 }
202 }
203 } while (token.number == tnull);
204
205 return token;
206
207
208 void lexicalError(int n) {
209     printf(" *** Lexical Error : ~");
210     switch (n) {
211     case 1: printf("an identifier length must be less than 12.\n");
212             break;
213     case 2: printf("next character must be &.\n");
214             break;
215     case 3: printf("next character must be !.\n");
216             break;
217     case 4: printf("Invalid character!!!\n");
```

```

main.c x
1025 (전역 범위) getIntNum(char first
208 void lexicalError(int n) {
209     printf(" *** Lexical Error : ");
210     switch(n) {
211         case 1: printf("an identifier length must be less than 12.\n");
212             break;
213         case 2: printf("next character must be &.\n");
214             break;
215         case 3: printf("next character must be !.\n");
216             break;
217         case 4: printf("invalid character!!!\n");
218             break;
219     }
220 }
221
222 int superLetter(char ch) {
223     if (isalpha(ch) || ch == '_') return 1;
224     else return 0;
225 }
226
227 int superLetterOrDigit(char ch) {
228     if (isalnum(ch) || ch == '_') return 1;
229     else return 0;
230 }
231
232 int getIntNum(char firstCharacter, FILE* source_file) {
233     int num = 0;
234     char ch;
235
236     ch = firstCharacter;
237     do {
238         num = 10 * num + (int)(ch - '0');
239         ch = fgetc(source_file);
240     } while (isdigit(ch));
241
242     ungetc(ch, stdin);
243     return num;
244 }

```

```

main.c x
1025 (전역 범위) lexicalError(int n)
245 void main(int argc, char *argv[]) {
246     FILE *source_file;
247     int i;
248     struct tokenType token;
249
250     if (argc != 2) {
251         fprintf(stderr, "Usage : scanner <source file name>\n");
252         exit(1);
253     }
254
255     if ((source_file = fopen(argv[1], "r")) == NULL) {
256         fprintf(stderr, "%s file not found\n", argv[1]);
257         exit(-1);
258     }
259
260     do {
261         for (i = 0; i < ID_LENGTH; i++)
262             id[i] = '\0';
263         token = scanner(source_file);
264         fprintf(stdout, "Token ----> ");
265
266         if (token.number == 5) { //상수
267             //for (i = 0; i < ID_LENGTH; i++)
268                 fprintf(stdout, "%d", token.value.num);
269             fprintf(stdout, " : (%d, %d)\n", token.number, token.value.num);
270         }
271         else if (token.number == 4) { //식별자
272             for (i = 0; i < ID_LENGTH; i++)
273                 fprintf(stdout, "%c", id[i]);
274             fprintf(stdout, " : (%d, %s)\n", token.number, token.value.id);
275         }
276         else { //지정어
277             if (isalpha(id[0])) {
278                 for (i = 0; i < ID_LENGTH; i++)
279                     fprintf(stdout, "%c", id[i]);
280                 fprintf(stdout, " : (%d, 0)\n", token.number);
281             }
282             else { //구분자, 연산자
283                 for (i = 0; i < ID_LENGTH; i++)
284                     fprintf(stdout, "%c", id[i]);
285                 fprintf(stdout, " : (%d, 0)\n", token.number);
286             }
287         }
288     } while (!feof(source_file));
289     fclose(source_file);
290 }
291

```


<실행결과>

VS 2017에 대한 x64_x86 Cross Tools 명령 프롬프트

```
C:\Users\2015305084\source\repos\1025\Debug>
C:\Users\2015305084\source\repos\1025\Debug>1025.exe prime.mc
Token --> const      : (30, 0)
Token --> int        : (33, 0)
Token --> max         : (4, max)
Token --> =           : (23, 0)
Token --> 100         : (5, 100)
Token --> ;           : (20, 0)
Token --> void        : (35, 0)
Token --> main        : (4, main)
Token --> (           : (7, 0)
Token --> )           : (8, 0)
Token --> {           : (37, 0)
Token --> int         : (33, 0)
Token --> i           : (4, i)
Token --> ;           : (14, 0)
Token --> j           : (4, j)
Token --> ;           : (14, 0)
Token --> k           : (4, k)
Token --> ;           : (20, 0)
Token --> int         : (33, 0)
Token --> rem         : (4, rem)
Token --> ,           : (14, 0)
Token --> prime       : (4, prime)
Token --> ;           : (20, 0)
Token --> i           : (4, i)
Token --> =           : (23, 0)
Token --> 2           : (5, 2)
Token --> ;           : (20, 0)
Token --> while       : (36, 0)
Token --> (           : (7, 0)
Token --> i           : (4, i)
Token --> <=          : (22, 0)
Token --> max         : (4, max)
Token --> )           : (8, 0)
Token --> {           : (37, 0)
Token --> prime       : (4, prime)
Token --> =           : (23, 0)
Token --> 1           : (5, 1)
Token --> ;           : (20, 0)
Token --> k           : (4, k)
Token --> =           : (23, 0)
Token --> i           : (4, i)
Token --> /           : (18, 0)
Token --> 2           : (5, 2)
Token --> ;           : (20, 0)
Token --> j           : (4, j)
Token --> =           : (23, 0)
Token --> 2           : (5, 2)
Token --> ;           : (20, 0)
Token --> while       : (36, 0)
```

VS 2017에 대한 x64_x86 Cross Tools 명령 프롬프트

```
Token --> 2           : (5, 2)
Token --> ;           : (20, 0)
Token --> while       : (36, 0)
Token --> (           : (7, 0)
Token --> j           : (4, j)
Token --> <=          : (22, 0)
Token --> k           : (4, k)
Token --> )           : (8, 0)
Token --> {           : (37, 0)
Token --> rem         : (4, rem)
Token --> =           : (23, 0)
Token --> i           : (4, i)
Token --> %           : (2, 0)
Token --> j           : (4, j)
Token --> ;           : (20, 0)
Token --> if          : (32, 0)
Token --> (           : (7, 0)
Token --> rem         : (4, rem)
Token --> ==          : (24, 0)
Token --> 0           : (5, 0)
Token --> )           : (8, 0)
Token --> prime       : (4, prime)
Token --> =           : (23, 0)
Token --> 0           : (5, 0)
Token --> ++          : (12, 0)
Token --> j           : (4, j)
Token --> ;           : (20, 0)
Token --> }           : (39, 0)
Token --> if          : (32, 0)
Token --> (           : (7, 0)
Token --> prime       : (4, prime)
Token --> ==          : (24, 0)
Token --> 1           : (5, 1)
Token --> )           : (8, 0)
Token --> write        : (4, write)
Token --> (           : (7, 0)
Token --> 1           : (5, 1)
Token --> )           : (8, 0)
Token --> ;           : (20, 0)
Token --> ++          : (12, 0)
Token --> i           : (4, i)
Token --> ;           : (20, 0)
Token --> }           : (39, 0)
Token --> }           : (39, 0)
Token --> >           : (29, 0)
```

C:\Users\2015305084\source\repos\1025\Debug>