

Project #1 Scanner Report

2022076062 김유찬

Project Goal

C - Minus scanner를 두 가지 방법으로 구현하기

- 방법 1 : 사용자 정의 C code 이용하기
 - DFA 방법으로 토큰 인식
 - scan.c를 수정하여 lexical rules 구현
- 방법 2 : Flex를 이용하여 어휘 패턴 지정
 - Regular Expression에 의해 lexical patterns 구체화
 - cminus.l 파일 수정하여 lexical rules 정의하기

Lexical Convention - 1

- Reserved words
int, void, if, else, while, return
- 19 symbols
+, -, *, /, <, ≤, >, ≥, ==, ≠, =, ::, (,), [,], {, }
- Identifier and number rules
ID = letter(letter | digit)*
NUM = digit digit*

Lexical Convention - 2

- Whitespaces
 - Spaces, newlines, tabs
 - 라인의 시작과 끝에 있는 공백은 무시하기
 - 문자, 숫자, 문자열 사이에 있는 공백은 토큰을 구분하는 데 사용하기
- Comments
 - 주석은 C 언어 표준의 형식을 따름 (/* */)
 - 한 줄 주석은 사용하지 않음 (//)
 - 주석은 중첩될 수 없음

custom C code 코드 수정

main.c

```
/* set NO_PARSE to TRUE to get a scanner-only compiler */  
#define NO_PARSE TRUE
```

NO_PARSE를 TRUE로 만들어 main에서 ENDFILE이 나올 때까지 getToken을 실행하여 토큰을 얻는다.

```
/* allocate and set tracing flags */  
int EchoSource = FALSE;  
int TraceScan = TRUE;
```

TraceScan을 TRUE로 만들어 scan.c에서 각 줄마다 토큰이 출력되도록 한다.

globals.h

```
/* MAXRESERVED = the number of reserved words */
#define MAXRESERVED 6

typedef enum
/* book-keeping tokens */
{ENDFILE,ERROR,
/* reserved words */
IF,ELSE,WHILE,RETURN,INT,VOID,
/* multicharacter tokens */
ID,NUM,
/* special symbols */
ASSIGN,EQ,NE,LT,LE,GT,GE,PLUS,MINUS,TIMES,OVER,LPAREN,RPAREN,LBRACE,RBRACE,LCURLY,RCURLY,SEMI,COMMA
} TokenType;
```

reserved words를 IF, ELSE, WHILE, RETURN, INT, VOID 6개로 바꿔주고

special symbol을 +, -, *, /, <, <=, >, >=, ==, !=, =, ;, ,, (,), [,], {, } 이것들로 맞춘다.

util.c

globals.h에 작성한 reserved words와 symbols를 기준으로 return 값 세팅한다.

세팅한 것과 다른 기호가 나오면 ERROR가 나오도록 처리한다.

scan.c

새롭게 추가된 enum StateType

```
/* states in scanner DFA */
typedef enum
{ START, INASSIGN, INCOMMENT, INNUM, INID, DONE, INNE, INLT, INGT }
StateType;
```

- INNE : '!'를 만났을 때 들어가는 상태, 이후 ERROR인지 != 인지 판단한다.
- INLT : '<'를 만났을 때 들어가는 상태, 이후 <인지 <=인지 판단한다.
- INGT : '>'를 만났을 때 들어가는 상태, 이후 >인지 >=인지 판단한다.

getToken 함수를 수정

INCOMMENT 처리

```
else if (c == '/') // 주석 시작 감지
{
    int next = getNextChar();
    if (next == '*'){ // '*' 이 나오면 주석임
        save = FALSE;
        state = INCOMMENT;
    }
    else{
        ungetNextChar(); // 주석이 아니면 나누기임
        currentToken = OVER;
        state = DONE;
    }
}
```

'/'를 입력 받으면 바로 다음에 '*'가 있는 지 없는 지 판단한다. '*'가 있다면 주석 처리 상태로 넘어가고 없다면 나누기 상태로 넘어간다.

```

case INCOMMENT:
    save = FALSE;
    if (c == EOF)
    { state = DONE;
      currentToken = ENDFILE;
    }
    else if (c == '*') { // 주석이 끝나는지 판단
      if (getNextChar() == '/') {
        state = START; // 처음으로 가서 다시 토큰 받기
      }
    }
  }
}

```

주석 상태에서는 주석이 끝나지 않았는데 EOF를 만나면 currentToken에 ENDFILE을 넣고 '/' 이후 부분을 모두 주석 처리한다. '*'가 들어왔을 때 다음 문자가 '/'일때만 state를 START를 넣어 다시 토큰을 받을 준비가 되는 상태로 만든다.

INID : ID = letter(letter|digit)*을 구현하기 위해 !isdigit(c) 조건 추가

INNE : '=' 이후에 '='이 있으면 NE 처리, 아니면 ERROR처리

INLT : '<' 이후에 '='이 있으면 LE 처리, 아니면 LT처리

INGT : '>' 이후에 '='이 있으면 GE처리, 아니면 GT처리

INASSIGN : '=' 이후에 '='이 또 있으면 EQ처리, 아니면 ASSIGN처리

cminus.l

definition section

```

letter      [a-zA-Z]
identifier  {letter}({letter}|{digit})*

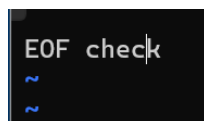
```

ID = ID = letter(letter|digit)*을 구현하기 위해 definition section에 identifier 정의 이후 rule section에 reserved words 와 symbols를 추가한다.

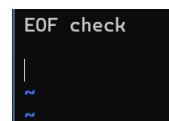
EOF에 대한 처리

입력에 대한 모든 것을 처리하고 입력이 없는 부분을 EOF 처리한다.

예)



EOF == 2



EOF == 4

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

```

11: reserved word: void
12: ID, name= main
13: {
14: reserved word: void
15: }
16: {
17: reserved word: int
18: ID, name= x
19: ;
20: reserved word: int
21: ID, name= y
22: ;
23: ;
24: ID, name= x
25: ;
26: ID, name= input
27: {
28: ;
29: ;
30: ID, name= y
31: ;
32: ID, name= input
33: {
34: ;
35: ;
36: ID, name= output
37: ;
38: ID, name= gcd
39: ;
40: ID, name= x
41: ;
42: ID, name= y
43: ;
44: ;
45: ;
46: ;
47: }
48: }
49: }
50: }
51: }
52: }
53: }
54: }
55: }
56: }
57: }
58: }
59: }
60: }
61: }
62: }
63: }
64: }
65: }
66: }
67: }
68: }
69: }
70: }
71: }
72: }
73: }
74: }
75: }
76: }
77: }
78: }
79: }
80: }
81: }
82: }
83: }
84: }
85: }
86: }
87: }
88: }
89: }
90: }
91: }
92: }
93: }
94: }
95: }
96: }
97: }
98: }
99: }
100: }
101: }
102: }
103: }
104: }
105: }
106: }
107: }
108: }
109: }
110: }
111: }
112: }
113: }
114: }
115: }
116: }
117: }
118: }
119: }
120: }
121: }
122: }
123: }
124: }
125: }
126: }
127: }
128: }
129: }
130: }
131: }
132: }
133: }
134: }
135: }
136: }
137: }
138: }
139: }
140: }
141: }
142: }
143: }
144: }
145: }
146: }
147: }
148: }
149: }
150: }
151: }
152: }
153: }
154: }
155: }
156: }
157: }
158: }
159: }
160: }
161: }
162: }
163: }
164: }
165: }
166: }
167: }
168: }
169: }
170: }
171: }
172: }
173: }
174: }
175: }
176: }
177: }
178: }
179: }
180: }
181: }
182: }
183: }
184: }
185: }
186: }
187: }
188: }
189: }
190: }
191: }
192: }
193: }
194: }
195: }
196: }
197: }
198: }
199: }
200: }
201: }
202: }
203: }
204: }
205: }
206: }
207: }
208: }
209: }
210: }
211: }
212: }
213: }
214: }
215: }
216: }
217: }
218: }
219: }
220: }
221: }
222: }
223: }
224: }
225: }
226: }
227: }
228: }
229: }
230: }
231: }
232: }
233: }
234: }
235: }
236: }
237: }
238: }
239: }
240: }
241: }
242: }
243: }
244: }
245: }
246: }
247: }
248: }
249: }
250: }
251: }
252: }
253: }
254: }
255: }
256: }
257: }
258: }
259: }
260: }
261: }
262: }
263: }
264: }
265: }
266: }
267: }
268: }
269: }
270: }
271: }
272: }
273: }
274: }
275: }
276: }
277: }
278: }
279: }
280: }
281: }
282: }
283: }
284: }
285: }
286: }
287: }
288: }
289: }
290: }
291: }
292: }
293: }
294: }
295: }
296: }
297: }
298: }
299: }
300: }
301: }
302: }
303: }
304: }
305: }
306: }
307: }
308: }
309: }
310: }
311: }
312: }
313: }
314: }
315: }
316: }
317: }
318: }
319: }
320: }
321: }
322: }
323: }
324: }
325: }
326: }
327: }
328: }
329: }
330: }
331: }
332: }
333: }
334: }
335: }
336: }
337: }
338: }
339: }
340: }
341: }
342: }
343: }
344: }
345: }
346: }
347: }
348: }
349: }
350: }
351: }
352: }
353: }
354: }
355: }
356: }
357: }
358: }
359: }
360: }
361: }
362: }
363: }
364: }
365: }
366: }
367: }
368: }
369: }
370: }
371: }
372: }
373: }
374: }
375: }
376: }
377: }
378: }
379: }
380: }
381: }
382: }
383: }
384: }
385: }
386: }
387: }
388: }
389: }
390: }
391: }
392: }
393: }
394: }
395: }
396: }
397: }
398: }
399: }
400: }
401: }
402: }
403: }
404: }
405: }
406: }
407: }
408: }
409: }
410: }
411: }
412: }
413: }
414: }
415: }
416: }
417: }
418: }
419: }
420: }
421: }
422: }
423: }
424: }
425: }
426: }
427: }
428: }
429: }
430: }
431: }
432: }
433: }
434: }
435: }
436: }
437: }
438: }
439: }
440: }
441: }
442: }
443: }
444: }
445: }
446: }
447: }
448: }
449: }
450: }
451: }
452: }
453: }
454: }
455: }
456: }
457: }
458: }
459: }
460: }
461: }
462: }
463: }
464: }
465: }
466: }
467: }
468: }
469: }
470: }
471: }
472: }
473: }
474: }
475: }
476: }
477: }
478: }
479: }
480: }
481: }
482: }
483: }
484: }
485: }
486: }
487: }
488: }
489: }
490: }
491: }
492: }
493: }
494: }
495: }
496: }
497: }
498: }
499: }
500: }
501: }
502: }
503: }
504: }
505: }
506: }
507: }
508: }
509: }
510: }
511: }
512: }
513: }
514: }
515: }
516: }
517: }
518: }
519: }
520: }
521: }
522: }
523: }
524: }
525: }
526: }
527: }
528: }
529: }
530: }
531: }
532: }
533: }
534: }
535: }
536: }
537: }
538: }
539: }
540: }
541: }
542: }
543: }
544: }
545: }
546: }
547: }
548: }
549: }
550: }
551: }
552: }
553: }
554: }
555: }
556: }
557: }
558: }
559: }
560: }
561: }
562: }
563: }
564: }
565: }
566: }
567: }
568: }
569: }
570: }
571: }
572: }
573: }
574: }
575: }
576: }
577: }
578: }
579: }
580: }
581: }
582: }
583: }
584: }
585: }
586: }
587: }
588: }
589: }
590: }
591: }
592: }
593: }
594: }
595: }
596: }
597: }
598: }
599: }
600: }
601: }
602: }
603: }
604: }
605: }
606: }
607: }
608: }
609: }
610: }
611: }
612: }
613: }
614: }
615: }
616: }
617: }
618: }
619: }
620: }
621: }
622: }
623: }
624: }
625: }
626: }
627: }
628: }
629: }
630: }
631: }
632: }
633: }
634: }
635: }
636: }
637: }
638: }
639: }
640: }
641: }
642: }
643: }
644: }
645: }
646: }
647: }
648: }
649: }
650: }
651: }
652: }
653: }
654: }
655: }
656: }
657: }
658: }
659: }
660: }
661: }
662: }
663: }
664: }
665: }
666: }
667: }
668: }
669: }
670: }
671: }
672: }
673: }
674: }
675: }
676: }
677: }
678: }
679: }
680: }
681: }
682: }
683: }
684: }
685: }
686: }
687: }
688: }
689: }
690: }
691: }
692: }
693: }
694: }
695: }
696: }
697: }
698:
```

[illegible]

```

11: reserved word: void
12: ID, name= main
13: (
14: reserved word: void
15: )
16: (
17: reserved word: int
18: ID, name= x
19: )
20: reserved word: int
21: ID, name= y
22: (
23: ID, name= x
24: =
25: ID, name= input
26: )
27: (
28: ID, name= y
29: )
30: ID, name= input
31: )
32: (
33: ID, name= output
34: =
35: ID, name= gcd
36: )
37: ID, name= x
38: =
39: ID, name= y
40: )
41: )
42: EOF

```

```
./cminus_cimpl test.2.txt
```

```
./cminus_lex test.2.txt
```

```
C:\msys64\usr\bin\media\sf_share\Project$ ./cmusncimpl test.2.txt
C-MUSIM COMPILATION: test.2.txt
1: reserved word: void
2: {
3:   ID, name= main
4:   {
5:     reserved word: void
6:   }
7: {
8:   reserved word: int
9:   ID, name= i
10:   {
11:     reserved word: int
12:     ID, name= x
13:   }
14:   NUM, val= 5
15: }
16: {
17:   ID, name= i
18:   =
19:   NUM, val= 0
20: }
21: {
22:   reserved word: while
23:   {
24:     ID, name= i
25:     <
26:     NUM, val= 5
27:   }
28: {
29:   ID, name= x
30: }
31: {
32:   ID, name= i
33: }
34: {
35:   ID, name= input
36: }
37: {
38:   }
39: }
40: ID, name= i
41: =
42: ID, name= i
43: *
44: NUM, val= 1
45: }
46: }
```

```

13: ID, name= i
14: =
15: NUM, val= 0
16:
17: reserved word: while
18: {
19:   ID, name= i
20:   ==
21:   NUM, val= 4
22: }
23: {
24:   reserved word: if
25:   {
26:     ID, name= x
27:     {
28:       ID, name= i
29:       {
30:         =
31:         NUM, val= 0
32:       }
33:       {
34:         ID, name= output
35:         {
36:           ID, name= x
37:           {
38:             ID, name= i
39:             {
40:               }
41:             }
42:           }
43:         }
44:       }
45:     }
46:   }
47: }
48: EOF

```

```

C-MINUS COMPILATION: test.2.txt
1: reserved word: void
2: {
3:   ID, name= main
4:   ;
5:   reserved word: void
6:   ;
7:   reserved word: int
8:   ID, name= i
9:   ;
10:  ;
11:  reserved word: int
12:  ID, name= x
13:  {
14:    NUM, val= 5
15:  }
16:  ;
17:  ;
18:  ID, name= i
19:  =
20:  NUM, val= 0
21:  ;
22:  reserved word: while
23:  {
24:    ID, name= i
25:    =
26:    NUM, val= 5
27:  }
28:  ;
29:  ID, name= input
30:  {
31:    ;
32:    ;
33:    ID, name= i
34:    =
35:    ID, name= i
36:    =
37:    NUM, val= 1
38:  }
39:  ;

```

```

13: ID, name= i
14: =
15: NUM, val= 0
16: ;
17: reserved word: while
18: (
19: ID, name= i
20: <=
21: NUM, val= 4
22: )
23: {
24: reserved word: if
25: (
26: ID, name= x
27: [
28: ID, name= i
29: ]
30: !=
31: NUM, val= 0
32: )
33: {
34: ID, name= output
35: (
36: ID, name= x
37: [
38: ID, name= i
39: ]
40: )
41: ;
42: ;
43: ;
44: }
45: }
46: }
47: EOF

```

!! : !=0 | 아니면 !, ! 따로 에러처리

'=' 배치에 따라 분리

reserved word 세로 형태와 가로형태

letter + digit 과 digit + letter

/* */이 한줄에 없음

주석 끝이 끝나지 않으면 주석 처리

EOF = 18

```

|!
= === ====
i
n
t
int a = 13;
letteranddigit = abc123
digitandletter = 123abc
/*
중간 주석 처리 잘 되나
*/
/* 주석이 끝나지 않으면
전부
주석
처리
~
~

```

./cminus_cimpl test.3.txt

```
muchan@compiler:/media/sf_share/Project1$ ./cminus_cimpl test.3.txt
C-MINUS COMPILATION: test.3.txt
1: ERROR: !
1: ERROR: !
2: =
2: ==
2: =
2: ==
2: ==
3: ID, name= i
4: ID, name= n
5: ID, name= t
6: reserved word: int
6: ID, name= a
6: =
6: NUM, val= 13
6: ;
7: ID, name= letteranddigit
7: =
7: ID, name= abc123
8: ID, name= digitandletter
8: =
8: NUM, val= 123
8: ID, name= abc
18: EOF
```

./cminus_lex test.3.txt

```
muchan@compiler:/media/sf_share/Project1$ ./cminus_lex test.3.txt
C-MINUS COMPILATION: test.3.txt
1: ERROR: !
1: ERROR: !
2: =
2: ==
2: =
2: ==
2: ==
3: ID, name= i
4: ID, name= n
5: ID, name= t
6: reserved word: int
6: ID, name= a
6: =
6: NUM, val= 13
6: ;
7: ID, name= letteranddigit
7: =
7: ID, name= abc123
8: ID, name= digitandletter
8: =
8: NUM, val= 123
8: ID, name= abc
18: EOF
```

난관

testcase를 메모장에서 작성하고 공유 폴더에 갖고와 저장을 하니 인코딩 형태가 utf-8 CRLF였다.

utf-8 CRLF는 줄바꿈이 \r\n 형태로 되어있어 왜인지는 모르겠지만 토큰 분리가 정확하지 않았다.

이후에 vim으로 작업하거나 메모장을 VSCode로 실행시켜 인코딩을 utf-8 LF로 바꿔주니 줄바꿈의 형태가 \n이 되었고 이후 정상적으로 작동했다.

줄 18, 열 1(60 선택됨) 공백: 4 UTF-8 LF 일반 텍스트 🔔

줄 18, 열 1(61 선택됨) 공백: 4 UTF-8 CRLF 일반 텍스트 🔔