Compiler Project

Project 1: Scanner

정상윤 (2016025032)

1. 컴파일 방식 & 환경

컴파일은 올려주신 Makefile 에서 -lfl 부분을 -ll 로 수정하고 그대로 사용하여 터미널에서 make 했습니다. Flex 설치는 homebrew를 통해 했습니다. 실습 환경은 macOS Big Sur (v 11.6) 에서 진행하였습니다.

2. 구현 방식 설명 & 실행 방법

프로젝트 설명 pdf 속 hint들을 많이 참고하면서 진행했습니다.

Methon 1 (DFA)

- **main.c**

NO_PARSE, TraceScan 을 TRUE로 설정해주었고, main 함수 안 “Tiny Compilation”을 “C-MINUS Compilation”으로 출력되도록 수정해줬습니다.

```
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;
```

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

- **globals.h**

기존의 keyword들을 IF, ELSE, WHILE, RETURN, INT, VOID 로 수정, 나머지는 삭제하였습니다. 또한 symbol들도 pdf hint 의 내용대로 수정했습니다.

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

- **util.c**

globals.h 의 수정 부분들에 맞춰서 printToken() 함수 내부 switch case들 수정, 각 symbol에 맞게 fprintf 출력 내용을 수정했습니다. ASSIGN은 ":="가 아닌 cminus 방식의 "="로 수정했습니다.

```
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 EQ: fprintf(listing,"==\n"); break;
28      case NE: fprintf(listing,"!=\n"); break;
29      case LT: 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 PLUS: fprintf(listing,"+\n"); break;
34      case MINUS: fprintf(listing,"-\n"); break;
35      case TIMES: fprintf(listing,"*\n"); break;
36      case OVER: fprintf(listing,"/\n"); break;
37      case LPAREN: fprintf(listing,"(\n"); break;
38      case RPAREN: fprintf(listing,")\n"); break;
39      case LBRACE: fprintf(listing,"[\n"); break;
40      case RBRACE: fprintf(listing,"]\n"); break;
41      case LCURLY: fprintf(listing,"{\n"); break;
42      case RCURLY: fprintf(listing,"}\n"); break;
43      case SEMI: fprintf(listing,";\n"); break;
44      case COMMA: fprintf(listing,",\n"); break;
```

- **scan.c**

우선 pdf hint 내용대로 reversedWords를 수정했습니다.

```
55  static struct
56      {
57        char* str;
58        TokenType tok;
59      } reservedWords[MAXRESERVED] = {
60        {"if",IF},
61        {"else",ELSE},
62        {"while",WHILE},
63        {"return",RETURN},
64        {"int",INT},
65        {"void",VOID}
66      };
```

StateType 속 state enum에 hint 대로 custom state들을 추가하였습니다. 그리고 기존의 사용하지 않을 INASSIGN은 삭제하였습니다.

```
13  typedef enum
14     { START,INCOMMENT,INNUM,INID,DONE,INEQ,INLT,INGT,INNE,INOVER,INCOMMENT_ }
15     StateType;
```

getToken() 함수 내부를 수정하는데 대부분의 시간을 썼습니다. 기존 코드들을 먼저 읽어보며 DFA가 어떻게 구현되는지 감을 잡은 후 수정을 진행했습니다. 대부분의 코드는 기존의 코드를 참고하여 작성했기에 세부적인 로직을 제외하고 대부분 비슷합니다. Space Bar, Tab, Enter 처리는 기존의 코드를 그대로 사용해도 됐습니다.
주석 처리를 어떻게 할지에 대해서 많은 생각을 했습니다. 우선 hint의 INCOMMENT 와 INCOMMENT_ 의 역할 차이가 뭔지 생각 후 INCOMMENT_ 는 INCOMMENT 상태에서 '*' 이 들어왔을 때를 처리해주기 위함이라는 것을 깨달은 후 코드를 작성했습니다. '/' 다음 '*'가 오면 주석, 그게 아니면 나눗셈이 되니 그 부분도 고려했습니다.

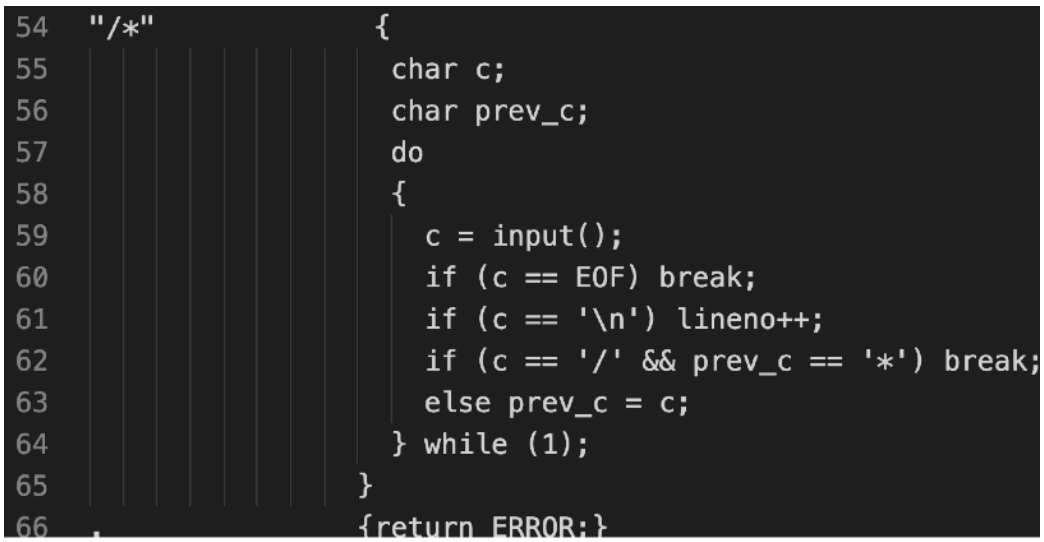
```
165         case INCOMMENT:
166           save = FALSE;
167           if (c == EOF)
168           {
169             state = DONE;
170             currentToken = ENDFILE;
171           }
172           // 주석 작성 중 '*' 가 들어오면 INCOMMENT_로
173           else if (c == '*') state = INCOMMENT_;
174           break;
```

```
175         case INCOMMENT_:
176           save = FALSE;
177           // "* -> /" 이면 주석 작성 끝
178           if(c == '/') state = START;
179           // EOF 처리
180           else if (c == EOF)
181           {
182             state = DONE;
183             currentToken = ENDFILE;
184           }
185           // "* -> /가 들어온게 아니면 여전히 주석 작성 중"
186           else state = INCOMMENT;
187           break;
```

Methon 2 (Regular Expression)

- **cminus.l**

큰 어려움은 없었던 것 같습니다. globals.h 와 마찬가지로 Keyword 들과 symbol 들을 수정해주고, 쓰이지 않을 것들은 삭제해줬습니다. 단, 주석 처리 관련해서 조금 신경을 써야 했습니다. "/*" 를 받은 상태에서 '*'가 들어온 후 다음 문자로 '/'로 들어오는 것을 검사해주는 로직이 필요했습니다. 처음에는 isStarWasInserted (int) 변수를 추가해줘서 주석을 벗어나는 조건을 검사해주려 했으나 예상한대로 잘 동작하지 못 해서 바로 전의 문자를 저장해주는 char형 변수를 추가해줘서 지금 문자가 '/'고 전 문자가 '*' 이면 주석이 끝나는 것으로 해주는 방식으로 고쳤고, 잘 동작하였습니다.



```
54  "/*"            {
55                   char c;
56                   char prev_c;
57                   do
58                   {
59                     c = input();
60                     if (c == EOF) break;
61                     if (c == '\n') lineno++;
62                     if (c == '/' && prev_c == '*') break;
63                     else prev_c = c;
64                   } while (1);
65                 }
66  .              {return ERROR;}
```

실행 방법

make clean -> make -> ./cminus_cimpl {input file} or ./cminus_lex {input file}

3. 실행 결과

Method 1 - test.1.txt 실행 결과

```
jeongsang-yun@jeongsang-yun-ui-MacBookAir 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 - test.2.txt 실행 결과

```
jeongsang-yun@jeongsang-yun-ui-MacBookAir 1_Scanner % ./cminus_lex test.2.txt

C-MINUS COMPILATION: test.2.txt
1: reserved word: void
1: ID, name= main
1: (
1: reserved word: void
1: )
2: {
3: reserved word: int
3: ID, name= i
3: ;
3: reserved word: int
3: ID, name= x
3: [
3: NUM, val= 5
3: ]
3: ;
5: ID, name= i
5: =
5: NUM, val= 0
5: ;
6: reserved word: while
6: (
6: ID, name= i
6: <
6: NUM, val= 5
6: )
7: {
8: ID, name= x
8: [
8: ID, name= i
8: ]
8: =
8: ID, name= input
8: (
8: )
8: ;
10: ID, name= i
10: =
10: ID, name= i
10: +
10: NUM, val= 1
10: ;
11: }
13: ID, name= i
13: =
13: NUM, val= 0
13: ;
14: reserved word: while
14: (
14: ID, name= i
14: <=
14: NUM, val= 4
14: )
15: {
16: reserved word: if
16: (
16: ID, name= x
16: [
16: ID, name= i
16: ]
16: !=
16: NUM, val= 0
16: )
17: {
18: ID, name= output
18: (
18: ID, name= x
18: [
18: ID, name= i
18: ]
18: )
18: ;
19: }
20: }
21: }
22: EOF
```