Project #1. Scanner

컴퓨터소프트웨어학부 2020079689 신다혜

1. Compilation environment

- Windows 11 Pro, Ubuntu 16.04 LTS
- Visual Studio Code 1.71.2
- gcc (Ubuntu 5.4.0-6ubuntu1~16.04.12) 5.4.0
- flex 2.6.0

2. Implementation

2-1. C code

- *globals.h*

  reserved words와 special symbols에 기존의 Tiny token을 지우고 C-Minus token을 추가한다.

- *util.c*

```c
void printToken( TokenType token, const char* tokenString )
{ switch (token)
  { case IF:
    case ELSE:
    case WHILE:
    case RETURN:
    case INT:
    case VOID:
      fprintf(listing,
         "reserved word: %s\n",tokenString);
      break;
    case ASSIGN: fprintf(listing,"=\n"); break;
    case LT: fprintf(listing,"<\n"); break;
    case LE: fprintf(listing, "<=\n"); break;
    case EQ: fprintf(listing,"==\n"); break;
    case NE: fprintf(listing, "!=\n"); break;
    case GT: fprintf(listing, ">\n"); break;
    case GE: fprintf(listing, ">=\n"); break;
    case LPAREN: fprintf(listing,"(\n"); break;
    case RPAREN: fprintf(listing,")\n"); break;
    case LBRACE: fprintf(listing, "[\n"); break;
    case RBRACE: fprintf(listing, "]\n"); break;
    case LCURLY: fprintf(listing, "{\n"); break;
    case RCURLY: fprintf(listing, "}\n"); break;
    case SEMI: fprintf(listing,";\n"); break;
    case COMMA: fprintf(listing, ",\n"); break;
    case PLUS: fprintf(listing,"+\n"); break;
    case MINUS: fprintf(listing,"-\n"); break;
    case TIMES: fprintf(listing,"*\n"); break;
    case OVER: fprintf(listing,"/\n"); break;
    case ENDFILE: fprintf(listing,"EOF\n"); break;
    case NUM:
      fprintf(listing,
         "NUM, val= %s\n",tokenString);
      break;
    case ID:
      fprintf(listing,
         "ID, name= %s\n",tokenString);
      break;
    case ERROR:
      fprintf(listing,
         "ERROR: %s\n",tokenString);
      break;
    default: /* should never happen */
      fprintf(listing,"Unknown token: %d\n",token);
  }
}
```

➔ C-Minus token에 맞도록 print 형태를 지정한다.

- *scan.c*

C-minus에 맞는 DFA를 구성하기 위해 기존의 Tiny token을 C-minus token으로 수정한다. getToken() 함수에서 input token을 식별하기 위해 state와 transition을 지정한다. 이 때 "==", "<=", ">="과 같이 여러 개의 문자로 이루어진 token은 한 번에 인식할 수 없으므로 INEQ, INLT, INRT 등 IN~ 형태의 예비 state를 만든다. comment의 경우 "/**/"를 차례대로 모두 찾아야하기 때문에 start state에서 "/"일 때 INOVER로, INOVER state에서 "*"일 때 INCOMMENT, INCOMMENT 에서 "*"일 때 INCOMMENT_ state로 옮겨간다.

```
240         case INOVER:
241           save = FALSE;
242           if (c == '*')
243           {
244             state = INCOMMENT;
245             tokenStringIndex--;
246           }
247           else
248           {
249             state = DONE;
250             ungetNextChar();
251             currentToken = OVER;
252           }
253           break;
254         case INCOMMENT:
255           save = FALSE;
256           if (c == EOF)
257           {
258             state = DONE;
259             currentToken = ENDFILE;
260           }
261           else if (c == '*') state = INCOMMENT_;
262           break;
263         case INCOMMENT_:
264           save = FALSE;
265           if (c == EOF)
266           {
267             state = DONE;
268             currentToken = ENDFILE;
269           }
270           else if (c == '*')
271             state = INCOMMENT_;
272           else if (c == '/')
273             state = START;
274           else
275             state = INCOMMENT;
276           break;
```

➔ INOVER, INCOMMENT, INCOMMENT_ state 부분

2-2. Lex (flex)

globals.h, main.c, util.c는 method1과 동일한 코드를 사용하고, getToken()은 Flex를 이용해서 자동으로 만들어지기 때문에 scan.c 파일은 사용되지않는다.

- *cminus.l*

Rules section에서 lexical pattern과 semantic action을 지정한다. C-minus token 각각에 맞는 action을 지정하며, "/*"는 아래의 규칙을 따른다.

```
54   "/*"            {
55                     char c, check;
56                     do
57                     {
58                       check = c;
59                       c = input();
60                       if (c == EOF) break;
61                       if (c == '\n') lineno++;
62                     } while (!(check == '*' && c == '/'));
63                   }
```

"*/"를 차례대로 찾아야하기 때문에 char형 변수 2개를 선언하고, check에 직전 input을 저장하며 c가 현재 input을 가져온다. 직전 input과 현재 input이 각각 "*"와 "/"일 때까지 do-while 문을 반복한다.

3. Run

```
$ make
$ (C code method) ./cminus_cimpl test.cm
$ (Lex method) ./cminus_lex test.cm
```

- test2.cm, test3.cm도 위 방법으로 동일하게 실행

4. Result

test.cm은 1_Scanner_2022_upload pdf파일에 제시된 example을 사용했으며 동일한 결과를 출력한다. 분량 관계상 test.cm은 생략하고 test2.cm과 test3.cm의 결과를 첨부한다.

4-1. test2.cm

- C-minus file

test2.cm

```
1   void main(void) {
2       /*
3       1
4       2 3     4
5       5 *********** 6
6       Comment not finished
7
8       int i; int j;
9       i=0;
10      j=2;
11      while(i<j)
12      {
13          i = (i+1);
14      }
15  }
16
```

Comment(/* */)가 끝나지 않는 test case

- Result

```
sdahye@ubuntu:~/Desktop/ele4029temp$ ./cminus_cimpl test2.cm

C-MINUS COMPILATION: test2.cm
        1: reserved word: void
        1: ID, name= main
        1: (
        1: reserved word: void
        1: )
        1: {
        16: EOF
```

```
sdahye@ubuntu:~/Desktop/ele4029temp$ ./cminus_lex test2.cm

C-MINUS COMPILATION: test2.cm
        1: reserved word: void
        1: ID, name= main
        1: (
        1: reserved word: void
        1: )
        1: {
        16: EOF
```

4-2. test3.cm

- C-minus file

test3.cm

```
1  void main(void) {
2      /*
3      1
4      2 3     4
5      5 *********** 6 */
6
7      int i; int j;
8      i=0;
9      j=0;
10     while(i<3)
11     {
12         i = input(; /* missing closing brace */
13         j = j + i;
14     }
15 }
16
```

line12에서 닫는 괄호가 없는 test case

- Result

```
sdahye@ubuntu:~/Desktop/ele4029temp$ ./cminus_cimpl test3.cm

C-MINUS COMPILATION: test3.cm
        1: reserved word: void
        1: ID, name= main
        1: (
        1: reserved word: void
        1: )
        1: {
        7: reserved word: int
        7: ID, name= i
        7: ;
        7: reserved word: int
        7: ID, name= j
        7: ;
        8: ID, name= i
        8: =
        8: NUM, val= 0
        8: ;
        9: ID, name= j
        9: =
        9: NUM, val= 0
        9: ;
        10: reserved word: while
        10: (
        10: ID, name= i
        10: <
        10: NUM, val= 3
        10: )
        11: {
        12: ID, name= i
        12: =
        12: ID, name= input
        12: (
        12: ;
        13: ID, name= j
        13: =
        13: ID, name= j
        13: +
        13: ID, name= i
        13: ;
        14: }
        15: }
        16: EOF
```

```
sdahye@ubuntu:~/Desktop/ele4029temp$ ./cminus_lex test3.cm

C-MINUS COMPILATION: test3.cm
        1: reserved word: void
        1: ID, name= main
        1: (
        1: reserved word: void
        1: )
        1: {
        7: reserved word: int
        7: ID, name= i
        7: ;
        7: reserved word: int
        7: ID, name= j
        7: ;
        8: ID, name= i
        8: =
        8: NUM, val= 0
        8: ;
        9: ID, name= j
        9: =
        9: NUM, val= 0
        9: ;
        10: reserved word: while
        10: (
        10: ID, name= i
        10: <
        10: NUM, val= 3
        10: )
        11: {
        12: ID, name= i
        12: =
        12: ID, name= input
        12: (
        12: ;
        13: ID, name= j
        13: =
        13: ID, name= j
        13: +
        13: ID, name= i
        13: ;
        14: }
        15: }
        16: EOF
```

test2, test3 모두 runtime error 없이 종료된다.