Scanner

2018009125 조성우

1. 코드구현 Method1 C implementation

```
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,COMM
   } TokenType;
```

Global.h에 있는 소스코드입니다.

기존의 몇몇 TokenType들을 지우고 요구사항에 맞게 새로운 TokenType들을 설정하였습니다.

```
static struct
    { char* str;
      TokenType tok;
    } reservedWords[MAXRESERVED]
   = {
        {"if",IF},
        {"else",ELSE},
        {"while",WHILE},
        {"return",RETURN},
        {"int",INT},
        {"void",VOID},
      };
```

Reserved words가 바뀌었기 때문에 MAXRESERVED도 6으로 바꾸었고 scan.c에있는reservedWords 구조체 배열의 내용도 바꾸었습니다.

util.c에 있는 printToken의 내용도 새로 설계한 TokenType에 맞게 각각의 내용들을 수정하였습니다.

이제부터는 scan.c의 코드입니다. 먼저 새로운 StateType을 새로 만들었습니다. StateType들은 getToken함수에서 현재의 state를 나타낼 때 사용됩니다.

```
                case '=':
                  state = INEQ;
                  break;
                case '!':
                  state = INNE;
                  break;
                case '<':
                  state = INLT;
                  break;
                case '>':
                  state = INGT;
                  break;
```

다음은 변수c가 =, !, <, >일 때의 상황을 나타내고 있습니다. 저 문자들은 symbol이 아직 정해지지 않았고 다음 문자에 따라 symbol정해집니다. 예를들어, =라는 문자를 입력 받았을 때 다음 문자가 =이면 '=='(EQ)를 가리키는 symbol이 되지만 다음 문자가 '='가 아니라면 '='(assign) symbol이 됩니다. '/'도 마찬가지로 다음문자가 '*'이면 comment가 되지만 '*'가 아니라면

단순히'/' symbol로 정의가 됩니다. 따라서 위의 문자들은 아직 symbol이 정해지지 않았기 때문에 state만 바꿔주었습니다.

```
165         case INEQ:
166           if (c == '=')
167             currentToken = EQ;
168           else{
169             if(c == EOF)lineno--;
170             ungetNextChar();
171             save = FALSE;
172             currentToken = ASSIGN;
173           }
174           state = DONE;
175           break;
```

그리고 이 변경된 state를 다음과 같이 getToken함수 내 다음번의 loop에서 사용합니다. State가 INEQ인 상태에서 새로운 문자 c를 입력받았고 이것이 '='이면 전 문자에 더해져 '=='EQ가 되지만 다른 문자라면 ungetNextChar()를 통해 문자하나를 무른다음 전에 입력받았던 c를 ASSIGN('=')라고 확정짓는 과정입니다. 169줄의 코드는 만약 분석하려는 소스코드의 마지막문자가 '='이고 바로 다음이 EOF일 때 lineno이 하나 더 커지는 상황을 방지하기 위한 것입니다. 처음에 '='를 입력받고 state가 INEQ로 바뀐다음 다시 getNextchar()를 호출하게 되는데 다음 문자는 없기 때문에 이 과정에서 lineno이 1증가하게 되고 c = EOF가 됩니다. 그리고 else문이 실행되고(이때 ungetNextchar는 lineno을 줄여주지 않습니다.) 마지막으로 '='을 출력하려고 하는데 lineno이 비정상적으로 1이 늘어난 상태에서 lineno이 출력되어버립니다. 그래서 이를 방지하고자 저런 코드를 삽입하였고 이 과정은 INEQ, INNE, INLT, INGT,INOVER에서도 동일합니다.

```
210         case INOVER:
211           if(c == '*'){
212             save = FALSE;
213             state = INCOMMENT;
214           }
215           else{
216             if(c == EOF)lineno--;
217             ungetNextChar();
218             save = FALSE;
219             currentToken = OVER;
220             state = DONE;
221           }
222           break;
223         case INCOMMENT:
224           save = FALSE;
225           if (c == EOF)
226           { state = DONE;
227             currentToken = ENDFILE;
228           }
229           else if(c == '*'){
230            state = INCOMMENT_;
231           }
232           break;
```

```
233         case INCOMMENT_:
234           save = FALSE;
235           if(c == '/') {
236            state = START;
237            tokenStringIndex = 0;
238           }
239           else if(c == EOF){
240            state = DONE;
241            currentToken = ENDFILE;
242           }
243           else if(c == '*') {
244            state = INCOMMENT_;
245           }
246           else state = INCOMMENT;
247           break;
```

다음은 INOVER, INCOMMENT와 INCOMMENT입니다. INOVER에서 '*'을 입력받으면 INCOMMENT로 넘어가고 INCOMMENT_는 INCOMMENT STATE일 때 '*'을 입력받으면 넘어가는 state입니다. 243줄에서 INCOMMENT_상태일 때 '*'을 입력받으면 계속 IMCOMMENT_상태를 유지하도록 하였습니다. '/'를 입력받으면 tokenStringIndex를 0으로 초기화하고 comment상황을 탈출하게 됩니다.

```
258         case INID:
259           if (!isalpha(c) && !isdigit(c))
260           { /* backup in the input */
261             if(c == EOF)lineno--;
262             ungetNextChar();
263             save = FALSE;
264             state = DONE;
265             currentToken = ID;
266           }
267           break;
```

마지막으로 INID입니다. 변수 첫 글자라 문자라면 그 뒤로는 숫자가 와도 상관없기에 !isdigit(c)을 조건문에 넣어주었습니다.

2. 코드구현 Method2 lex implementation

```
19    identifier  {letter}({letter}|{digit})*
```

identifier에 첫글자가 문자가 나오면 숫자가 나와도 괜찮기 때문에 다음과 같이 바꿔주었습니다.

```
32    "!="          {return NE;}
33    "<"           {return LT;}
34    "=="          {return EQ;}
35    "<="          {return LE;}
36    ">"           {return GT;}
37    ">="          {return GE;}
```

!=, ==, <=, >= 등은 다음과 같이 설계하였습니다.

```
54    "/*"          { char c;
55                    char prev;
56                      while(1){
57                      c = input();
58                      while(c == '*'){
59                        prev = c;
60                        c = input();
61                      }
62                      if(c == '/' && prev == '*')break;
63                      if (c == '\0'){
64                        if(prev != '\n')lineno++;
65                        break;
66                      }
67                      if (c == '\n') lineno++;
68                      prev = c;
69                      }
70                  }
```

comment입니다 comment상태일 때 '*'을 입력받는다면 inner while문을 통해 계속 반복시켰습니다. 이 때 '*'이 아닌 문자가 입력되면 loop를 탈출하는데 그 문자가 '/'였다면 break를 통해 comment를 벗어나게 하였습니다. (이때 prev는 항상 '*') 63줄에서 EOF를 만났을 때 prev가 '₩n'이 아니라면 lineno++을 하여 다음 라인에 EOF가 찍히도록 하였습니다. 이는 test2의 상황에서 result2와 같은 결과값이 나오도록 인위적으로 바꾼 것입니다.

3. Test result

cimpl과 lex가 동일한 결과가 나왔기 때문에 cimpl의 결과만을 첨부하겠습니다.

```
C-MINUS COMPILATION: ./Example/test2.cm
       1: ID, name= A
       3: /
       3: *
       4: ID, name= comment
       4: ID, name= test
       4: NUM, val= 2
       6: *
       6: /
       8: ID, name= A
       16: ID, name= B
       21: EOF
```

```
C-MINUS COMPILATION: ./Example/test3.cm
       1: reserved word: return
       2: reserved word: void
       3: reserved word: while
       4: ID, name= until
       5: ID, name= write
       6: ID, name= read
       7: ID, name= end
       8: ==
       9: !=
       10: =
       11: ;
       12: ,
       13: (
       14: EOF
```

```
C-MINUS COMPILATION: ./Example/test4.cm
       1: ID, name= ab
       2: ID, name= abcd
       3: ID, name= a13
       4: NUM, val= 1
       5: NUM, val= 123
       6: ERROR: !
       7: !=
       8: EOF
```

왼쪽부터 차례대로 test2.cm test3.cm test4.cm 의 결과입니다. 지면 부족으로 test.cm와 test5.cm은 첨부하지 못했지만 모든 case에서 result예시와 동일한 결과가 나왔습니다.