1. Scanner

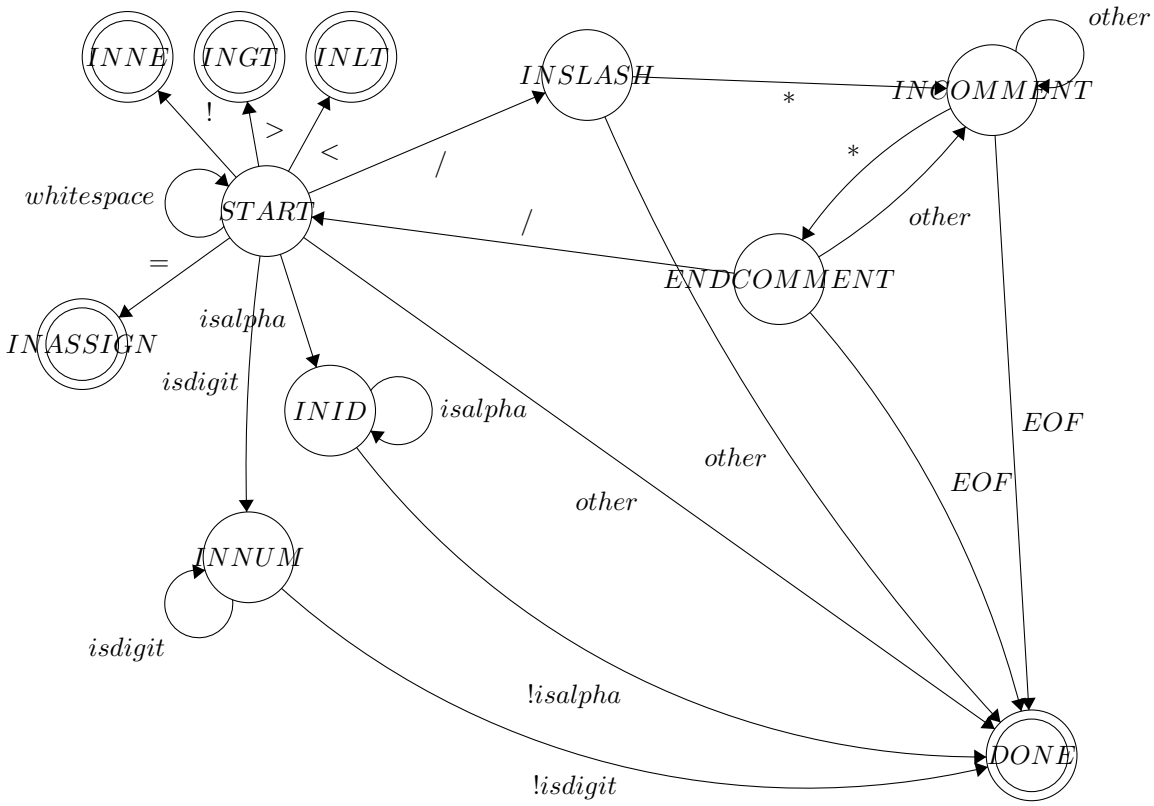
2019027192 김현수

2021년 10월 31일

차 례

1 cimpl

getToken 함수에서 state의 변화를 나타내는 DFA는 아래 그림과 같습니다. INASSIGN/INNE/INGT/INLT는 다음에 나오는 문자에 상관없이 항상 DONE으로 이동하지만, 이에 대한 연결선을 그림에 넣으면 그림이 너무 복잡해져 보고서 공간에 들어가지 않았기 때문에 accept 노드로 표현했습니다.



위 DFA의 accept 노드들은 적절한 조건에 따라 currentToken, linepos를 변화시킵니다. 이에 대한 간단한 설명은 아래에서 이어집니다.

A) INNE

'='가 등장한다면 "!="가 나온 것이므로 currentToken을 NE로 합니다. '='를 제외한 문자가 등장한다면 c- 문법에서 지원하지 않는 경우가 나온 것이므로, currentToken을 ERROR로 합니다.

B) INGT

'='가 등장한다면 "¿="가 나온 것이므로 currentToken을 GE로 합니다. '='을 제외한 문자가 등장한다면 "¿"만 나온 것이므로, currentToken을 GT로 하고 ungetNextChar을 호출해 linepos를 1 감소시킵니다.

C) INLT

'='가 등장한다면 "¡="가 나온 것이므로 currentToken을 LE로 합니다. '='을 제외한 문자가 등장한다면 "¡"만 나온 것이므로, currentToken을 LT로 하고 ungetNextChar을 호출해 linepos를 1 감소시킵니다.

D) INASSIGN

'='가 등장한다면 "=="가 나온 것이므로 currentToken을 EQ로 합니다. '='을 제외한 문자가 등장한다면 "="만 나온 것이므로, currentToken을 ASSIGN으으로 하고 ungetNextChar을 호출해 linepos를 1 감소시킵니다.

E) ENDCOMMENT

'/'가 등장한다면 "*""가 나온 것이므로 state를 START로 변경합니다. EOF가 등장한다면 주석이 열린 상태로 파일이 종료된 것이므로, currentToken을 ENDFILE로 해 scan이 끝났음을 알립니다. 이 둘을 제외한 문자가 등장한다면 단순히 주석 내에서 * 문자가 등장한 것이므로, 다시 INCOMMENT state로 되돌아갑니다.

F) INSLASH

가 등장할 경우 "/" 이므로 state를 INCOMMENT로 변화시킵니다. '*'을 제외한 문자가 등장한다면 "/"만 나온 것이므로, currentToken을 OVER로 하고 ungetNextChar을 호출해 linepos를 1 감소시킵니다.

G) INCOMMENT

INCOMMENT state에서는 "*/"이나 EOF가 등장하기 전까지 loop를 돕니다. '*'가 등장하면 state를 ENDCOMMENT로 변화시킵니다. EOF가 등장하면 currentToken을 ENDFILE로, state를 DONE으로 변화시켜 더 scan할 내용이 없다는 것을 알립니다.

2 lex

c-의 lex 구현은 ./lex/tiny.l 파일을 조금 수정해서 할 수 있습니다. 먼저 아래와 같이 c-에 존재하는 패턴들을 입력합니다. digit/number/letter/identifier/newline/whitespace에 대한 규칙들은 tiny와 c-가 동일하기 때문에 변경하지 않아도 됩니다.

```
"if"            {return IF;}
"else"          {return ELSE;}
"while"         {return WHILE;}
"return"        {return RETURN;}
"int"           {return INT;}
"void"          {return VOID;}
"="             {return ASSIGN;}
"=="            {return EQ;}
"!="            {return NE;}
"<"             {return LT;}
"<="            {return LE;}
">"             {return GT;}
">="            {return GE;}
"+"             {return PLUS;}
"-"             {return MINUS;}
"*"             {return TIMES;}
"/"             {return OVER;}
"("             {return LPAREN;}
")"             {return RPAREN;}
"["             {return LBRACE;}
"]"             {return RBRACE;}
"{"             {return LCURLY;}
"}"             {return RCURLY;}
";"             {return SEMI;}
","             {return COMMA;}
```

다음으로 주석에 대한 패턴을 입력합니다. c-에서 주석은 "/*"부터 "*/"까지로 정의됩니다. 따라서 아래 코드와 같이 구현해줍니다. '*'가 등장하면 end_comment를 TRUE로 합니다. 그리고 end_comment가 TRUE인 상태에서 '/'가 등장하면 "*/"가 나온 것이므로 break를 통해 주석을 끝내줍니다. 반대로 end_comment가 TRUE인 상태에서 '/'가 아닌 문자가 등장한다면 end_comment = c == '*'; 조건문에 따라 end_comment를 다시 설정해줍니다.

```
"/*"            { char c;
                  int end_comment = FALSE;
                  do
                  { c = input();
                    if (c == EOF) break;
                    if (end_comment && c == '/') break;
                    if (c == '\n') lineno++;
                    end_comment = c == '*';
                  } while (TRUE);
                }
```

3 compilation environment

Ubuntu 20.04 LTS

gcc (Ubuntu 9.3.0-17ubuntu1 20.04) 9.3.0

flex 2.6.4

1_Scanner 폴더 내에서 make 실행하면 cminus_cimpl, cminus_lex 파일이 생성됩니다.

```
 dodo@dodo ~/github/2021_ele4029_2019027192/1_Scanner  master  make
gcc  -W -Wall -c -o main.o main.c
main.c:48:1: warning: return type defaults to ‘int’ [-Wimplicit-int]
   48 | main(int argc, char* argv[])
      | ^~~~
main.c: In function ‘main’:
main.c:50:15: warning: unused variable ‘syntaxTree’ [-Wunused-variable]
   50 |     TreeNode* syntaxTree;
      |               ^~~~~~~~~~
gcc  -W -Wall -c -o util.o util.c
util.c:165:8: warning: type defaults to ‘int’ in declaration of ‘indentno’ [-Wimplicit-int]
  165 | static indentno = 0;
      |        ^~~~~~~~
gcc  -W -Wall -c -o scan.o scan.c
gcc  -W -Wall -o cminus_cimpl main.o util.o scan.o
flex -o lex.yy.c cminus.l
gcc  -W -Wall -c -o lex.yy.o lex.yy.c
lex.yy.c:1275:17: warning: ‘yyunput’ defined but not used [-Wunused-function]
 1275 |     static void yyunput (int c, char * yy_bp )
      |                 ^~~~~~~~
gcc  -W -Wall -o cminus_lex main.o util.o lex.yy.o -lfl
 dodo@dodo ~/github/2021_ele4029_2019027192/1_Scanner  master  ls ¦ grep cminus_
cminus_cimpl
cminus_lex
```

test.cm 파일을 인자로 넣어 테스트 해 보면, 같은 결과물이 나오는 것을 알 수 있습니다.

```
dodo@dodo ~/github/2021_ele4029_2019027192/1_Scanner  master  ./cminus_cimpl test.cm

TINY COMPILATION: test.cm
       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
```

```
dodo@dodo ~/github/2021_ele4029_2019027192/1_Scanner  master  ./cminus_lex test.cm

TINY COMPILATION: test.cm
       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
```