Compiler Scanner Report

학부	컴퓨터소프트웨어학부
학번	2018008977
이름	이종범

Environment

Compiler: clang 14.0.3

Flex: 2.6.4 Apple(flex-34)

OS: Ventura 13.5.2

Device: Apple MacBook Pro M1 Max 32GB

How to Run?

- Compile

  $ make all

- C Code Implementation

  $ ./cminus_cimpl [C-Minus File to Run]

- Lex(Flex) Implementation

  $ ./cminus_lex [C-Minus File to Run]

C Code Implementation

1. globals.h 파일 내 Reserved Word 목록 및 상수 변경

   C-Minus의 Reserved Word의 내용과 동일하게 Enum 및 상수를 <그림 1>과 같이 변경

```
25  25  /* MAXRESERVED = the number of reserved words */
26      -#define MAXRESERVED 8
    26  +#define MAXRESERVED 6
27  27
28  28  typedef enum
29  29      /* book-keeping tokens */
30  30     {ENDFILE,ERROR,
31  31      /* reserved words */
32      -    IF,THEN,ELSE,END,REPEAT,UNTIL,READ,WRITE,
    32  +    IF,THEN,ELSE,WHILE,RETURN,INT,VOID,
33  33      /* multicharacter tokens */
34  34      ID,NUM,
35  35      /* special symbols */
36      -    ASSIGN,EQ,LT,PLUS,MINUS,TIMES,OVER,LPAREN,RPAREN,SEMI
    36  +    ASSIGN,EQ,NE,LT,LE,GT,GE,PLUS,MINUS,TIMES,OVER,LPAREN,RPAREN,LBRACE,RBRACE,LCURLY,RCURLY,SEMI,COMMA
37  37     } TokenType;
```

<그림 1> globals.h 파일 내 Reserved Word 목록 및 상수 변경 사항

2. scan.c 파일 내 Reserved Word Lookup 테이블 수정

   Token의 Reserved Word 여부를 판단하는 Lookup 테이블을 C-Minus의 것과 동일하게 <그림 2>와 같이 변경

```
54  54  /* lookup table of reserved words */
55  55  static struct
56  56      { char* str;
57  57        TokenType tok;
58  58      } reservedWords[MAXRESERVED]
59      -   = {{"if",IF},{"then",THEN},{"else",ELSE},{"end",END},
60      -      {"repeat",REPEAT},{"until",UNTIL},{"read",READ},
61      -      {"write",WRITE}};
    59  +   = {{"if",IF},{"else",ELSE},{"while",WHILE},{"return",RETURN},{"int",INT},{"void",VOID}};
```

<그림 2> scan.c 파일 내 Reserved Word Lookup 테이블 변경 사항

3. util.c 파일 내 printToken 함수의 출력 가능 토큰 목록을 C-Minus의 것과 동일하게 변경

   출력 가능 Token 목록을 C-Minus의 것과 동일하게 <그림 3>, <그림 4>와 같이 변경

```
26      -    case ASSIGN: fprintf(listing,":=\n"); break;
    26  +    case ASSIGN: fprintf(listing,"=\n"); break;
    27  +    case EQ: fprintf(listing,"==\n"); break;
    28  +    case NE: fprintf(listing,"!=\n"); break;
27  29      case LT: fprintf(listing,"<\n"); break;
28      -    case EQ: fprintf(listing,"=\n"); break;
29      -    case LPAREN: fprintf(listing,"(\n"); break;
30      -    case RPAREN: fprintf(listing,")\n"); break;
31      -    case SEMI: 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;
32  33      case PLUS: fprintf(listing,"+\n"); break;
```

<그림 3> util.c 파일 내 printToken 변경 사항 중 일부 1

```
33  34      case MINUS: fprintf(listing,"-\n"); break;
34  35      case TIMES: fprintf(listing,"*\n"); break;
35  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;
```

<그림 4> util.c 파일 내 printToken 변경 사항 중 일부 1

4. scan.c 파일 내 DFA State Enum 변경

과제에 제공된 Hint와 동일하게 DFA State를 담는 StateType Enum을 <그림 5>와 같이 변경

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

<그림 5> scan.c 파일 내 StateType Enum 변경 사항

5. scan.c 파일 내 getToken 함수에 C-Minus Scanner DFA 구현

I. 기존 INASSIGN State 관련 코드 제거

C-Minus에서는 Assign Symbol이 '='이므로 기존 ':='에 대한 처리를 <그림 6>, <그림 7>과 같이 제거

```
95      -          else if (c == ':')
96      -            state = INASSIGN;
```

<그림 6> INASSIGN State 제거

```
151     -      case INASSIGN:
152     -        state = DONE;
153     -        if (c == '=')
154     -          currentToken = ASSIGN;
155     -        else
156     -        { /* backup in the input */
157     -          ungetNextChar();
158     -          save = FALSE;
159     -          currentToken = ERROR;
160     -        }
161     -        break;
```

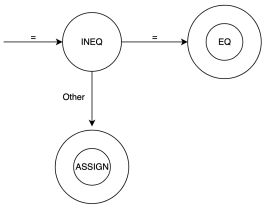
<그림 7> INASSIGN State 에서의 Transition 관련 코드 제거

II. '=', '==', '!=' Symbol에 대한 State 구현

<그림 8>과 같이 =을 만나면 INEQ State에 진입하고 이후 입력되는 글자에 따라 EQ, ASSIGN 여부를 <그림 9>의 코드와 같이 판단

INEQ State 다음 읽히는 문자가 =가 아닌 경우에는 ASSIGN 토큰으로 처리하고, 추가적으로 읽힌 문자를 ungetNextChar 함수를 호출하여 되돌림

'!=' Symbol도 이와 동일하게 구현되었으나, Other 문자가 입력되었을 때 ERROR Token으로 인식된다는 것이 다름



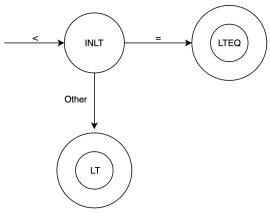
<그림 8> '='과 '==' State 처리를 위한 DFA

```
97  +          else if (c == '=')
98  +            state = INEQ;
165 +      case INEQ:
166 +        if (c == '=')
167 +        { state = DONE;
168 +          currentToken = EQ;
169 +        }
170 +        else
171 +        { /* backup in the input */
172 +          ungetNextChar();
173 +          save = FALSE;
174 +          state = DONE;
175 +          currentToken = ASSIGN;
176 +        }
```

<그림 9> '=', '==' 관련 DFA 구현 코드

III. '<', '<=', '>', '>=' Symbol에 대한 State 구현

<그림 10>과 <그림 11>과 같이 '<', '<=' 관련 DFA를 구현하고 '>', '>='도 이와 동일하게 구현



<그림 10> '<'과 '<=' State 처리를 위한 DFA

```
99  +          else if (c == '<')
100 +            state = INLT;
177 +      case INLT:
178 +        if (c == '=')
179 +        { state = DONE;
180 +          currentToken = LE;
181 +        }
182 +        else
183 +        { /* backup in the input */
184 +          ungetNextChar();
185 +          save = FALSE;
186 +          state = DONE;
187 +          currentToken = LT;
188 +        }
```

<그림 11> '<', '<=' 관련 DFA 구현 코드

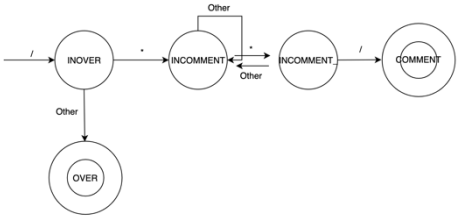
IV. 단일 Character Symbol('[', '(', ...)에 대한 State 구현

해당 문자가 입력되었을 때 즉시 Token으로 반환할 수 있도록 함

V. Identifier 관련 State 구현

기존 Tiny Compiler에서는 문자로 이루어진 것만 ID로 인식하므로 INID State에서 Transition 판단 로직 내에 'isdigit(c)'와 같이 함수를 호출하여 숫자도 ID 내에 인식할 수 있도록 수정

VI. Comment, '/' 관련 State 구현



<그림 12> Comment, '/' 관련 DFA

<그림 12>와 같이 DFA를 구성하여 구현

'/' 문자를 만나면 INOVER State에 진입하고, 다음 문자에 따라 OVER 또는 INCOMMENT State로 전이함

INCOMMENT State에서는 '*/'를 만날 때까지 INCOMMENT State를 벗어나지 않음

Lex(Flex) Implementation

1. cminus.l 파일 내 Definition Section 변경

   기존 Tiny Compiler에서는 문자로 이루어진 것만 ID로 인식하므로 identifier의 Regular Expression을 <그림 13>과 같이 변경하여 identifier가 letter(letter | digit)*를 인식하도록 수정

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

<그림 13> identifier Regular Expression 수정

2. cminus.l 파일 내 Rules Section 변경

   해당 Scanner가 C-Minus의 Symbol을 읽을 수 있도록 <그림 14>, <그림 15>와 같이 수정

```
      25  +"int"          {return INT;}
      26  +"void"         {return VOID;}
25    27   "if"           {return IF;}
26        -"then"         {return THEN;}
27    28   "else"         {return ELSE;}
28        -"end"          {return END;}
29        -"repeat"       {return REPEAT;}
30        -"until"        {return UNTIL;}
31        -"read"         {return READ;}
32        -"write"        {return WRITE;}
33        -":="           {return ASSIGN;}
34        -"="            {return EQ;}
35        -"<"            {return LT;}
      29  +"while"        {return WHILE;}
      30  +"return"       {return RETURN;}
```

<그림 14> C-Minus Symbol 추가 1

```
      35  +"<"            {return LT;}
      36  +"<="           {return LE;}
      37  +">"            {return GT;}
      38  +">="           {return GE;}
      39  +"=="           {return EQ;}
      40  +"!="           {return NE;}
      41  +"="            {return ASSIGN;}
      42  +";"            {return SEMI;}
      43  +","            {return COMMA;}
40    44   "("            {return LPAREN;}
41    45   ")"            {return RPAREN;}
42        -";"            {return SEMI;}
      46  +"["            {return LBRACE;}
      47  +"]"            {return RBRACE;}
      48  +"{"            {return LCURLY;}
      49  +"}"            {return RCURLY;}
```

<그림 15> C-Minus Symbol 추가 2

Comment의 경우 위와 다르게 /*가 입력되었을 경우 */를 만날 때까지 사이의 모든 입력을 무시할 수 있도록 <그림 16>과 같이 구현함

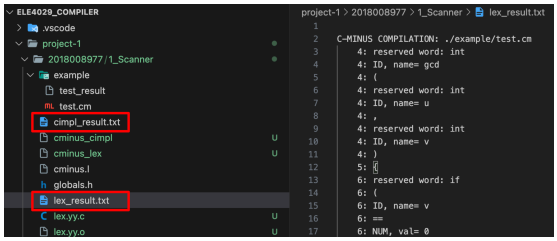
```
47        -"{"            { char c;
      54  +"/*"           { char c;
      55  +                 int state = 0;
48    56                    do
49    57                    { c = input();
50    58                      if (c == EOF) break;
51    59                      if (c == '\n') lineno++;
52        -                   } while (c != '}');
      60  +                   if (c == '*') state = 1;
      61  +                   else if (c == '/' && state == 1) break;
      62  +                   else state = 0;
      63  +                 } while (1);
53    64                    }
```

<그림 16> Comment Rules Section

Result

제공된 test.cm 파일을 예시 파일로 사용하고, 각각 cimpl_result.txt, lex_result.txt로 저장하여 결과물을 제공된 test_result 파일과 비교함

- $ make all
- $ ./cminus_cimpl ./example/test.cm > ./cimpl_result.txt
- $ ./cminus_lex ./example/test.cm > ./lex_result.txt



- $ diff ./cimpl_result.txt ./lex_result.txt

```
1_Scanner % diff ./cimpl_result.txt ./lex_result.txt
1_Scanner %
```

차이점 없음

- $ diff ./example/test_result ./lex_result.txt

```
leejongbeom@JBMacBookPro 1_Scanner % diff ./example/test_result ./lex_result.txt
2c2
< C-MINUS COMPILATION: test.cm
---
> C-MINUS COMPILATION: ./example/test.cm
```

파일 경로 외에 차이점 없음을 확인