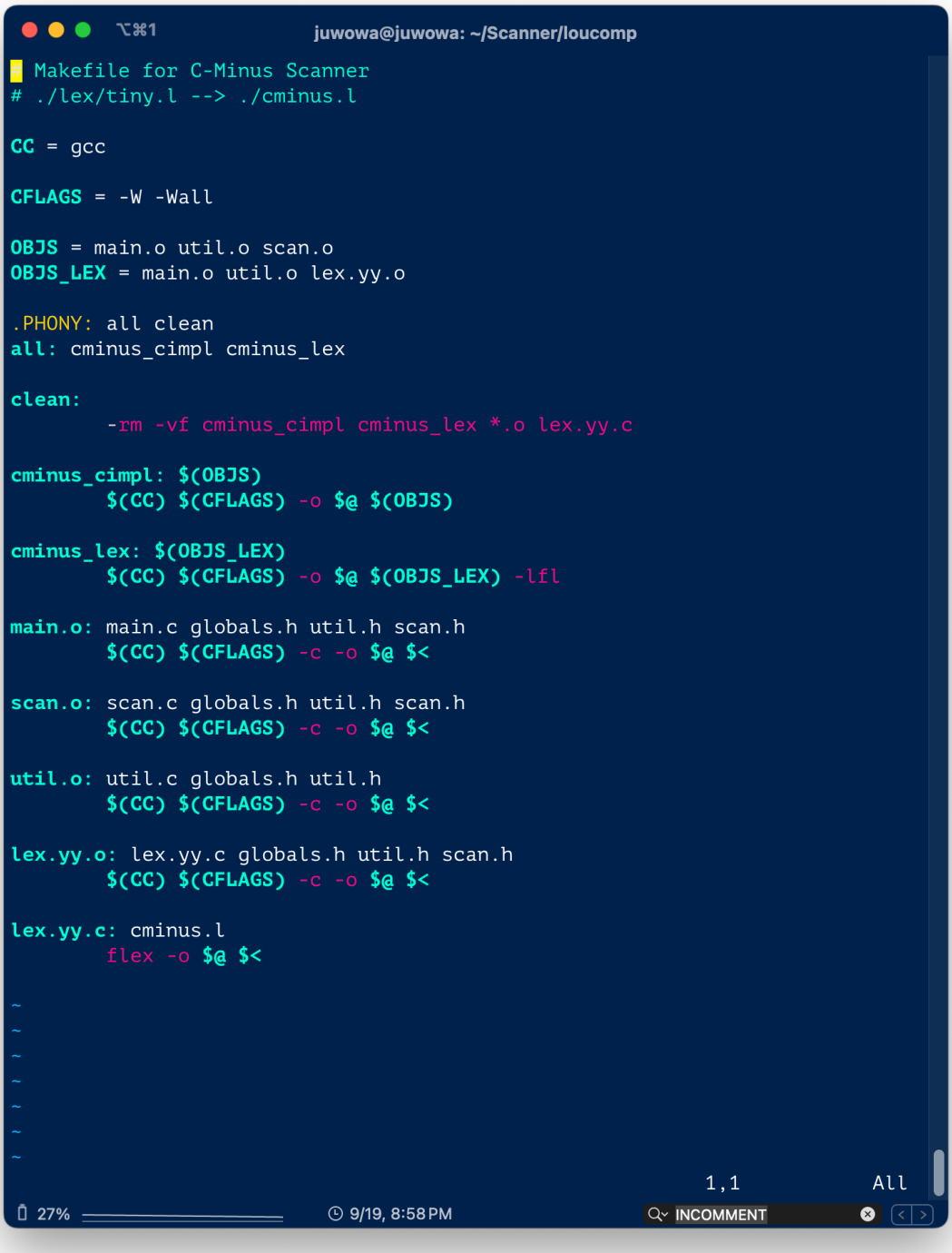
Compiler_Project1_Scanner

2019060546

주원웅

1. 컴파일



2. 개발 환경

- Ubuntu 22.04

3. 구현과 작동

이번 프로젝트에서는 공통적으로 main.c, globals.h, utils.c 의 일부 코드를 수정한 후 C code 구현 방식 에서는 scan.c 를 수정하고, Lex 방식에서는 cminus.l 파일을 생성하여 C-Minus Scanner를 구현하였다.

- 1) Main.c 수정부분

```
#define NO_PARSE TRUE
int TraceScan = TRUE;
```

- 2) Scan.c 수정부분

```
/* states in scanner DFA */
typedef enum
{
  START,
  INASSIGN,
  INCOMMENT,
  INCOMMENT_,
  INEQ,
  INLT,
  INGT,
  INNE,
  INOVER,
  INNUM,
  INID,
  DONE
} StateType;
```

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

```
/* lookup an identifier to see if it is a reserved word */
/* uses linear search */
static TokenType reservedLookup(char *s)
{
  int i;
  for (i = 0; i < MAXRESERVED; i++)
    if (!strcmp(s, reservedWords[i].str))
      return reservedWords[i].tok;
  return ID;
}
```

Scan.c 의 MAXRESERVED 의 범위를 수정해주기 위해 globals.h 를 수정했다. 그렇지 않으면 일반 ID(identifier) 를 식별할 때 reservedWords 배열을 확인하는 과정에서 segfault 가 발생하여 프로그램 에러가 발생하였다.

- 3) Globals.h 수정부분

```
/* MAXRESERVED = the number of reserved words */
#define MAXRESERVED 6
```

```
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,
   COMMA
} TokenType;
```

사용 할 토큰들을 추가해주었고, Scan.c 에서 사용할 MAXRESERVED 길이를 수정해주었다.

- 4) Util.c 수정부분

```
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 EQ: fprintf(listing,"==\n"); break;
    case NE: fprintf(listing,"!=\n"); break;

    case LT: fprintf(listing,"<\n"); break;
    case LE: 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;
```

스캐너의 결과로 출력될 부분인 util.c 에 추가된 토큰의 출력을 추가했고, 기존 토큰의 출력을 수정했다

- 5) Cminus.l 수정부분

```
"/*"            {
                  int c;
                  int comment_done = 0;
                  while (!comment_done) {
                    c = input();  // 주석 안의 문자 하나씩 읽기
                    if (c == EOF)
                      break;
                    if (c == '\n') {
                      lineno++;  // 새로운 줄일 때 라인 번호 증가
                    }
                    if (c == '*') {  // 주석 끝을 찾기 위해 '*'를 체크
                      c = input();
                      if (c == '/') {
                        comment_done = 1;  // 주석 종료
                      }
                    }
                  }
                }
```

```
"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;}
```

- 토큰들을 추가해주었고, 주석처리 부분을 수정하여 cminus.l 파일을 완성하였다.

실행결과

1. cimnus_cimpl 결과

```
juwowa@juwowa:~/Scanner/loucomp$ ./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: int
        6: ID, name= ss
        6: =
        6: ID, name= u
        6: +
        6: ID, name= v
        6: ;
        7: reserved word: if
        7: (
        7: ID, name= v
```

```
    6: ID, name= v
    6: ;
    7: reserved word: if
    7: (
    7: ID, name= v
    7: ==
    7: NUM, val= 0
    7: )
    7: reserved word: return
    7: ID, name= u
    7: ;
    8: reserved word: else
    8: reserved word: return
    8: ID, name= gcd
    8: (
    8: ID, name= v
    8: ,
    8: ID, name= u
    8: -
    8: ID, name= u
    8: /
    8: ID, name= v
    8: *
    8: ID, name= v
    8: )
    8: ;
    10: }
    12: reserved word: int
    12: ID, name= sss
    12: (
    12: reserved word: int
    12: ID, name= as
    12: ,
    12: reserved word: int
    12: ID, name= bd
    12: )
    13: {
    14: reserved word: if
    14: (
    14: ID, name= as
    14: >=
    14: ID, name= bd
    14: )
    14: reserved word: return
    14: NUM, val= 2
    14: ;
    15: reserved word: else
    15: reserved word: return
    15: ID, name= as
    15: ;
```

```
    14: ;
    15: reserved word: else
    15: reserved word: return
    15: ID, name= as
    15: ;
    17: reserved word: while
    17: (
    17: ID, name= as
    17: >
    17: NUM, val= 0
    17: )
    17: {
    17: ID, name= arr
    17: [
    17: ID, name= i
    17: ]
    17: -
    17: -
    17: }
    18: reserved word: if
    18: (
    18: ID, name= as
    18: ==
    18: ID, name= bd
    18: )
    18: {
    18: }
    19: reserved word: if
    19: (
    19: ID, name= bd
    19: <
    19: ID, name= as
    19: )
    19: {
    19: }
    20: reserved word: if
    20: (
    20: ID, name= bd
    20: <=
    20: ID, name= as
    20: )
    20: {
    20: }
    21: reserved word: if
    21: (
    21: ID, name= as
    21: !=
    21: ID, name= bd
    21: )
    21: {
```

```
    21: ID, name= bd
    21: )
    21: {
    21: }
    22: reserved word: if
    23: }
    25: reserved word: void
    25: ID, name= boy
    25: (
    25: )
    25: {
    25: reserved word: return
    25: ;
    25: }
    27: reserved word: void
    27: ID, name= main
    27: (
    27: reserved word: void
    27: )
    28: {
    29: reserved word: int
    29: ID, name= x
    29: ;
    29: reserved word: int
    29: ID, name= y
    29: ;
    30: ID, name= x
    30: =
    30: ID, name= input
    30: (
    30: )
    30: ;
    30: ID, name= y
    30: =
    30: ID, name= input
    30: (
    30: )
    30: ;
    31: ID, name= output
    31: (
    31: ID, name= gcd
    31: (
    31: ID, name= x
    31: ,
    31: ID, name= y
    31: )
    31: )
    31: ;
    32: }
    35: EOF
```

2. cimnus_lex 결과

```
juwowa@juwowa:~/Scanner/loucomp$ ./cminus_lex 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: int
        6: ID, name= ss
        6: =
        6: ID, name= u
        6: +
        6: ID, name= v
        6: ;
        7: reserved word: if
```

```
    6: ID, name= u
    6: +
    6: ID, name= v
    6: ;
    7: reserved word: if
    7: (
    7: ID, name= v
    7: ==
    7: NUM, val= 0
    7: )
    7: reserved word: return
    7: ID, name= u
    7: ;
    8: reserved word: else
    8: reserved word: return
    8: ID, name= gcd
    8: (
    8: ID, name= v
    8: ,
    8: ID, name= u
    8: -
    8: ID, name= u
    8: /
    8: ID, name= v
    8: *
    8: ID, name= v
    8: )
    8: ;
    10: }
    12: reserved word: int
    12: ID, name= sss
    12: (
    12: reserved word: int
    12: ID, name= as
    12: ,
    12: reserved word: int
    12: ID, name= bd
    12: )
    13: {
    14: reserved word: if
    14: (
    14: ID, name= as
    14: >=
    14: ID, name= bd
    14: )
    14: reserved word: return
    14: NUM, val= 2
    14: ;
    15: reserved word: else
    15: reserved word: return
```

```
15: reserved word: return
15: ID, name= as
15: ;
17: reserved word: while
17: (
17: ID, name= as
17: >
17: NUM, val= 0
17: )
17: {
17: ID, name= arr
17: [
17: ID, name= i
17: ]
17: -
17: -
17: }
18: reserved word: if
18: (
18: ID, name= as
18: ==
18: ID, name= bd
18: )
18: {
18: }
19: reserved word: if
19: (
19: ID, name= bd
19: <
19: ID, name= as
19: )
19: {
19: }
20: reserved word: if
20: (
20: ID, name= bd
20: <=
20: ID, name= as
20: )
20: {
20: }
21: reserved word: if
21: (
21: ID, name= as
21: !=
21: ID, name= bd
21: )
21: {
21: }
22: reserved word: if
```

```
21: )
21: {
21: }
22: reserved word: if
23: }
25: reserved word: void
25: ID, name= boy
25: (
25: )
25: {
25: reserved word: return
25: ;
25: }
27: reserved word: void
27: ID, name= main
27: (
27: reserved word: void
27: )
28: {
29: reserved word: int
29: ID, name= x
29: ;
29: reserved word: int
29: ID, name= y
29: ;
30: ID, name= x
30: =
30: ID, name= input
30: (
30: )
30: ;
30: ID, name= y
30: =
30: ID, name= input
30: (
30: )
30: ;
31: ID, name= output
31: (
31: ID, name= gcd
31: (
31: ID, name= x
31: ,
31: ID, name= y
31: )
31: )
31: ;
32: }
35: EOF
```