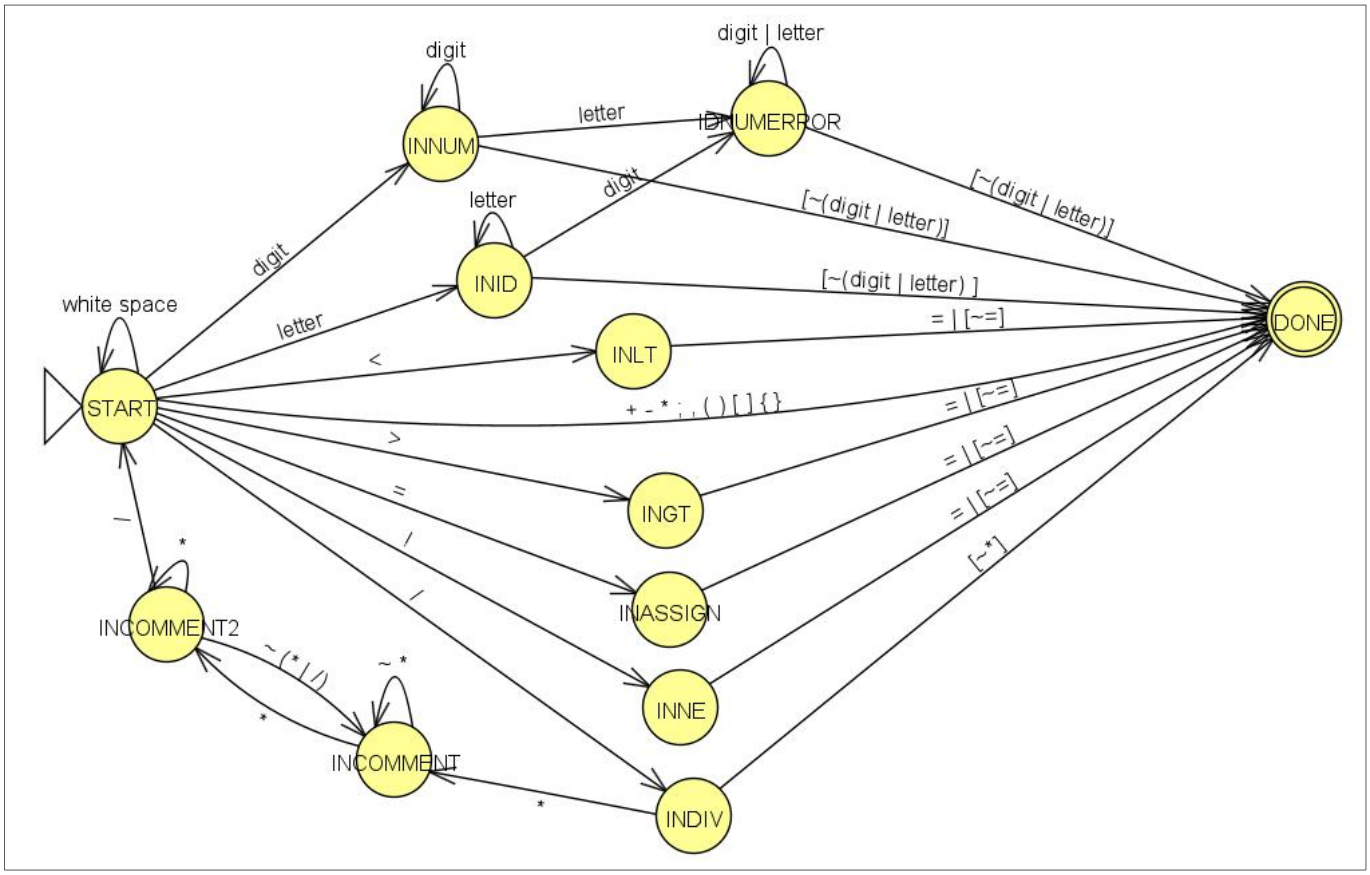
# C- 언어 스캐너 구현 보고서

github.com/squaryun

## 1. DFA




## 2. Scanning 과정

1번에서 구성한 DFA를 이용하여 다음 예시 문장을 스캐닝하는 과정을 나타내고자 합니다.

<예시 문장>

```
if (score!  =200)/* this includes error **/ num=98F;
```

| Char | Transition Funtion | Output | Description |
|---|---|---|---|
| i | T(START, 'i') = INID | | 'i'를 읽고 INID로 전이 |
| f | T(INID, 'f') = INID | | 'f'를 읽고 전이 X |
| ' ' | T(INID, ' ') = DONE | reserved word: if | ' '를 읽고 DONE으로 전이<br>' '는 Lookahead로 소비하지 않음<br>토큰은 ID(식별자)로 구분하지만,<br>ID가 예약어 테이블에 있음을 확인 |
| ' ' | T(START, ' ') = START | | ' '를 읽고 전이 X |
| ( | T(START, '(') = DONE | ( | '('를 읽고 DONE으로 전이<br>토큰은 LPAREN으로 구분 후 출력 |

| | | | |
|---|---|---|---|
| s | T(START, 's') = INID | | 's'를 읽고 INID로 전이 |
| c | T(INID, 'c') = INID | | 'c'를 읽고 전이 X |
| o | T(INID, 'o') = INID | | 'o'를 읽고 전이 X |
| r | T(INID, 'r') = INID | | 'r'를 읽고 전이 X |
| e | T(INID, 'e') = INID | | 'e'를 읽고 전이 X |
| ! | T(INID, '!') = DONE | ID, name= score | '!'를 읽고 DONE으로 전이<br>'!'는 Lookahead로 소비하지 않음<br>토큰은 ID(식별자)로 구분 후 출력 |
| ! | T(START, '!') = INNE | | '!'를 읽고 INNE로 전이 |
| ' ' | T(INNE, ' ') = DONE | ERROR: ! | ' '를 읽고 DONE으로 전이<br>' '는 Lookahead로 소비하지 않음<br>토큰은 ERROR으로 구분 후 출력 |
| ' ' | T(START, ' ') = START | | ' '를 읽고 전이 X |
| ' ' | T(START, ' ') = START | | ' '를 읽고 전이 X |
| = | T(START, '=') = INASSIGN | | '='를 읽고 INASSIGN으로 전이 |
| 2 | T(INASSIGN, '2') = DONE | = | '2'를 읽고 DONE으로 전이<br>'2'는 Lookahead로 소비하지 않음<br>토큰은 EQ으로 구분 후 출력 |
| 2 | T(START, '2') = INNUM | | '2'를 읽고 INNUM으로 전이 |
| 0 | T(INNUM, '0') = INNUM | | '0'를 읽고 전이 X |
| 0 | T(INNUM, '0') = INNUM | | '0'를 읽고 전이 X |
| ) | T(INNUM, ')') = DONE | NUM, val= 200 | ')'를 읽고 DONE으로 전이<br>')'는 Lookahead로 소비하지 않음<br>토큰은 NUM으로 구분 후 출력 |
| ) | T(START, ')') = DONE | ) | ')'를 읽고 DONE으로 전이<br>토큰은 RPAREN으로 구분 후 출력 |
| ' ' | T(START, ' ') = START | | ' '를 읽고 전이 X |
| / | T(START, '/') = INDIV | | '/'를 읽고 INDIV로 전이 |
| * | T(INDIV, '*') = INCOMMENT | | '*'를 읽고 INCOMMENT로 전이 |
| ' ' | T(INCOMMENT, ' ') = INCOMMENT | | ' '를 읽고 전이 X |
| 't' 'h' 'i' 's' ' ' 'i' 'n' 'c' 'l' 'u' 'd' 'e' 's' ' ' 'e' 'r' 'r' 'o' 'r' ' '를 읽고 전이 X | | | |
| * | T(INCOMMENT, '*') = INCOMMENT2 | | '*'를 읽고 INCOMMENT2로 전이 |
| * | T(INCOMMENT2, '*') = INCOMMENT2 | | '*'를 읽고 전이 X |
| / | T(INCOMMENT2, '/') = START | | '*'를 읽고 START로 전이 |
| ' ' | T(START, ' ') = START | | ' '를 읽고 전이 X |
| n | T(START, 'n') = INID | | 'n'를 읽고 INID로 전이 |
| u | T(INID, 'u') = INID | | 'u'를 읽고 전이 X |
| m | T(INID, 'm') = INID | | 'm'를 읽고 전이 X |
| ' ' | T(INID, ' ') = DONE | ID, name= num | ' '를 읽고 DONE으로 전이<br>' '는 Lookahead로 소비하지 않음<br>토큰은 ID으로 구분 후 출력 |
| ' ' | T(START, ' ') = START | | ' '를 읽고 전이 X |
| = | T(START, '=') = INASSIGN | | '='를 읽고 INASSIGN으로 전이 |
| ' ' | T(INASSIGN, ' ') = DONE | = | ' '를 읽고 DONE으로 전이<br>' '는 Lookahead로 소비하지 않음<br>토큰은 ASSIGN으로 구분 후 출력 |
| ' ' | T(START, ' ') = START | | ' '를 읽고 전이 X |
| 9 | T(START, '9') = INNUM | | '0'를 읽고 INNUM으로 전이 |
| 8 | T(INNUM, '8') = INNUM | | '8'를 읽고 전이 X |

| | | | |
|---|---|---|---|
| F | T(INNUM, 'F') = IDNUMERROR | | 'F'를 읽고 IDNUMERROR로 전이 |
| ; | T(IDNUMERROR, ';') = DONE | ERROR: 98F | ';'를 읽고 DONE으로 전이<br>';'는 Lookahead로 소비하지 않음<br>토큰은 ERROR으로 구분 후 출력 |
| ; | T(START, ';') = DONE | ; | ';'를 읽고 DONE으로 전이<br>토큰은 SEMI으로 구분 후 출력 |

## 3. 구현 설명

저는 교재에 있는 두 번째 방법(Better method)을 이용하여 구현하였습니다. 강의 자료 코드 scan.c를 수정하여 직접 작성하였습니다. 구현 과정을 https://www.github.com/squareyun 에 업로드 하였으니 참고해주시면 감사하겠습니다. 소스 코드 중 중요한 역할을 하는 함수인 getToken() 함수를 첨부합니다.

```
TokenType getToken(void) {
	int tokenStringIndex = 0;			// 토큰 문자열(tokenString)의 index
	TokenType currentToken = STARTFILE;	// 현재 토큰
	StateType state = START;			// 시작 state는 항상 START
	int save;					// tokenString에 토큰을 저장할지 확인하는 flag
	while (state != DONE)	// 토큰이 DONE이 아닐 때 까지 반복
	{
		int c = getNextChar();  // 다음 character 읽어오기
		save = TRUE;
		switch (state)
		{
		case START:
			if ((c == ' ') || (c == '\t') || (c == '\n'))
				save = FALSE;
			else if (isdigit(c))
				state = INNUM;
			else if (isalpha(c))
				state = INID;
			else if (c == '<')
				state = INLT;
			else if (c == '>')
				state = INGT;
			else if (c == '=')
				state = INASSIGN;
			else if (c == '/')
				state = INDIV;
			else if (c == '!')
				state = INNE;
			else
			{
				state = DONE;
				switch (c)
```

```
                        {
                        case EOF:
                                save = FALSE;
                                currentToken = ENDFILE;
                                break;
                        case '+':
                                currentToken = PLUS;
                                break;
                        case '-':
                                currentToken = MINUS;
                                break;
                        case '*':
                                currentToken = MUL;
                                break;
                        case ';':
                                currentToken = SEMI;
                                break;
                        case ',':
                                currentToken = COMMA;
                                break;
                        case '(':
                                currentToken = LPAREN;
                                break;
                        case ')':
                                currentToken = RPAREN;
                                break;
                        case '[':
                                currentToken = LSQUARE;
                                break;
                        case ']':
                                currentToken = RSQUARE;
                                break;
                        case '{':
                                currentToken = LCURLY;
                                break;
                        case '}':
                                currentToken = RCURLY;
                                break;
                        default: // 에러토큰
                                currentToken = ERROR;
                                break;
                        }
                }
                break;
        case INNUM:
```

```
                // digit을 읽으면 전이 X
                // digit과 letter가 붙는 경우 에러토큰으로 인식 (e.g., 111aaa)
                if (isalpha(c))
                        state = IDNUMERROR;
                else if (!isdigit(c))
                {
                        state = DONE;
                        ungetNextChar(); // Lookahead. 문자를 소모하지 않고 되돌리는 함수
                        save = FALSE;
                        currentToken = NUM;
                }
                break;
        case IDNUMERROR:
                // digit 또는 letter을 읽으면 전이 X
                if (!isalpha(c) && !isdigit(c)) {
                        state = DONE;
                        ungetNextChar();        // Lookahead
                        save = FALSE;
                        currentToken = ERROR;
                }
                break;
        case INID:
                // letter을 읽으면 전이 X
                // letter과 digit이 붙는 경우 에러 토큰으로 인식 (e.g., aaa111)
                if (isdigit(c))
                        state = IDNUMERROR;
                else if (!isalpha(c))
                {
                        ungetNextChar();        // Lookahead
                        save = FALSE;
                        state = DONE;
                        currentToken = ID;
                }
                break;
        case INLT:
                state = DONE;
                if (c == '=')   // 토큰 '<='을 인식
                        currentToken = LE;
                else
                {
                        ungetNextChar();        // Lookahead
                        save = FALSE;
                        currentToken = LT;
                }
                break;
```

```
		case INGT:
			state = DONE;
			if (c == '=')   // 토큰 '>='을 인식
				currentToken = GE;
			else
			{
				ungetNextChar();        // Lookahead
				save = FALSE;
				currentToken = GT;
			}
			break;
		case INASSIGN:
			state = DONE;
			if (c == '=')   // 토큰 '=='을 인식
				currentToken = EQ;
			else
			{
				ungetNextChar();        // Lookahead
				save = FALSE;
				currentToken = ASSIGN;
			}
			break;
		case INNE:
			state = DONE;
			if (c == '=')   // 토큰 '!='을 인식
				currentToken = NE;
			else
			{
				ungetNextChar();        // Lookahead
				save = FALSE;
				currentToken = ERROR;
			}
			break;
		case INDIV:
			if (c == '*') {         // '/*' 으로 comment의 시작을 인식
				save = FALSE;   // comment는 save 하지 않음
				state = INCOMMENT;
			}
			else {
				ungetNextChar();        // Lookahead
				state = DONE;
				currentToken = DIV;
			}
			break;
		case INCOMMENT:
```

```
                        // *이 아닌 character를 읽으면 전이 X
                        tokenStringIndex = 0; // tokenString에 이미 저장된 문자(/)를 무시
                        save = FALSE;
                        if (c == EOF) { // non-final state에서 프로그램이 종료되면 에러 메세지 출력
                                fprintf(fpOut, "ERROR: %s\n", "\"stop before ending\"");
                                exit(EXIT_FAILURE);
                        }
                        else if (c == '*')
                                state = INCOMMENT2;
                        break;
                case INCOMMENT2:
                        // *를 읽으면 전의 X
                        save = FALSE;
                        if (c == '/')
                                state = START;
                        else if (c != '*')
                                state = INCOMMENT;
                        break;
                case DONE:
                default: /* should never happen */
                        fprintf(fpOut, "Scanner Bug: state= %d\n", state);
                        state = DONE;
                        currentToken = ERROR;
                        break;
                }
                if ((save) && (tokenStringIndex <= MAXTOKENLEN))
                        tokenString[tokenStringIndex++] = (char)c; // 토큰 문자열에 character 추가
                if (state == DONE)
                {
                        tokenString[tokenStringIndex] = '\0';
                        if (currentToken == ID)
                                // ID(식별자)가 예약어 테이블에 있는지 확인.
                                // 있으면 해당 예약어 토큰이 반환됨
                                currentToken = reservedLookup(tokenString);
                }
        }
        fprintf(fpOut, "\t%d: ", lineno);
        printToken(currentToken, tokenString);

        return currentToken;
}
```

## 4. 실행 결과

sample 프로그램을 돌렸을 때의 실행 결과는 다음과 같습니다.

### 4.1. sample code 1.c

```
C- COMPILATION: 1.c
    1: /* A program to perform Euclid's
    2: Algorithm to compute gcd */
    3:
    4: int gcd (int u, int v)
           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: {    if (v==0) return u;
           5: {
           5: reserved word: if
           5: (
           5: ID, name= v
           5: ==
           5: NUM, val= 0
           5: )
           5: reserved word: return
           5: ID, name= u
           5: ;
    6:       else return gcd(v, u-u/v*v);
           6: reserved word: else
           6: reserved word: return
           6: ID, name= gcd
           6: (
           6: ID, name= v
           6: ,
           6: ID, name= u
           6: -
           6: ID, name= u
           6: /
           6: ID, name= v
           6: *
           6: ID, name= v
           6: )
```

```
        6: ;
 7:       /* u-u/v*v == u mod v */
 8: }
        8: }
 9:
10: void main(void)
        10: reserved word: void
        10: ID, name= main
        10: (
        10: reserved word: void
        10: )
11: {    int x; int y;
        11: {
        11: reserved word: int
        11: ID, name= x
        11: ;
        11: reserved word: int
        11: ID, name= y
        11: ;
12:       x=input(); y=input();
        12: ID, name= x
        12: =
        12: ID, name= input
        12: (
        12: )
        12: ;
        12: ID, name= y
        12: =
        12: ID, name= input
        12: (
        12: )
        12: ;
13:       output(gcd(x,y));
        13: ID, name= output
        13: (
        13: ID, name= gcd
        13: (
        13: ID, name= x
        13: ,
        13: ID, name= y
        13: )
        13: )
        13: ;
14: }
        14: }
```

```
 15:
        16: EOF
```

### 4.2. sample code 2.c

```
C- COMPILATION: 2.c
    1: /* A program to perform selection sort on a 10
    2:      element array. */
    3:
    4: int x[10];
            4: reserved word: int
            4: ID, name= x
            4: [
            4: NUM, val= 10
            4: ]
            4: ;
    5:
    6: int minloc ( int a[], int low, int high )
            6: reserved word: int
            6: ID, name= minloc
            6: (
            6: reserved word: int
            6: ID, name= a
            6: [
            6: ]
            6: ,
            6: reserved word: int
            6: ID, name= low
            6: ,
            6: reserved word: int
            6: ID, name= high
            6: )
    7: {  int i; int x; int k;
            7: {
            7: reserved word: int
            7: ID, name= i
            7: ;
            7: reserved word: int
            7: ID, name= x
            7: ;
            7: reserved word: int
            7: ID, name= k
            7: ;
```

```
 8:    k = low;
       8: ID, name= k
       8: =
       8: ID, name= low
       8: ;
 9:    x = a[low];
       9: ID, name= x
       9: =
       9: ID, name= a
       9: [
       9: ID, name= low
       9: ]
       9: ;
10:    i = low + 1;
       10: ID, name= i
       10: =
       10: ID, name= low
       10: +
       10: NUM, val= 1
       10: ;
11:    while (i < high)
       11: reserved word: while
       11: (
       11: ID, name= i
       11: <
       11: ID, name= high
       11: )
12:    {          if (a[i] < x)
       12: {
       12: reserved word: if
       12: (
       12: ID, name= a
       12: [
       12: ID, name= i
       12: ]
       12: <
       12: ID, name= x
       12: )
13:                        { x = a[i];
       13: {
       13: ID, name= x
       13: =
       13: ID, name= a
       13: [
       13: ID, name= i
```

```
        13: ]
        13: ;
14:                         k = i;   }
        14: ID, name= k
        14: =
        14: ID, name= i
        14: ;
        14: }
15:             i = i + 1;
        15: ID, name= i
        15: =
        15: ID, name= i
        15: +
        15: NUM, val= 1
        15: ;
16:    }
        16: }
17:    return k;
        17: reserved word: return
        17: ID, name= k
        17: ;
18: }
        18: }
19:
20: void sort( int a[], int low, int high)
        20: reserved word: void
        20: ID, name= sort
        20: (
        20: reserved word: int
        20: ID, name= a
        20: [
        20: ]
        20: ,
        20: reserved word: int
        20: ID, name= low
        20: ,
        20: reserved word: int
        20: ID, name= high
        20: )
21: { int i; int k;
        21: {
        21: reserved word: int
        21: ID, name= i
        21: ;
        21: reserved word: int
```

```
        21: ID, name= k
        21: ;
22:     i = low;
        22: ID, name= i
        22: =
        22: ID, name= low
        22: ;
23:
24:     while (i < high-1)
        24: reserved word: while
        24: (
        24: ID, name= i
        24: <
        24: ID, name= high
        24: -
        24: NUM, val= 1
        24: )
25:     {           int t;
        25: {
        25: reserved word: int
        25: ID, name= t
        25: ;
26:             k = minloc(a,i,high,i);
        26: ID, name= k
        26: =
        26: ID, name= minloc
        26: (
        26: ID, name= a
        26: ,
        26: ID, name= i
        26: ,
        26: ID, name= high
        26: ,
        26: ID, name= i
        26: )
        26: ;
27:             t = a[k];
        27: ID, name= t
        27: =
        27: ID, name= a
        27: [
        27: ID, name= k
        27: ]
        27: ;
28:             a[k] = a[i];
```

```
        28: ID, name= a
        28: [
        28: ID, name= k
        28: ]
        28: =
        28: ID, name= a
        28: [
        28: ID, name= i
        28: ]
        28: ;
29:             a[i] = t;
        29: ID, name= a
        29: [
        29: ID, name= i
        29: ]
        29: =
        29: ID, name= t
        29: ;
30:             i = i + 1;
        30: ID, name= i
        30: =
        30: ID, name= i
        30: +
        30: NUM, val= 1
        30: ;
31:     }
        31: }
32: }
        32: }
33:
34: void main(void)
        34: reserved word: void
        34: ID, name= main
        34: (
        34: reserved word: void
        34: )
35: { int i;
        35: {
        35: reserved word: int
        35: ID, name= i
        35: ;
36:     i = 0;
        36: ID, name= i
        36: =
        36: NUM, val= 0
```

```
        36: ;
37:     while (i < 10)
        37: reserved word: while
        37: (
        37: ID, name= i
        37: <
        37: NUM, val= 10
        37: )
38:     {           x[i] = input();
        38: {
        38: ID, name= x
        38: [
        38: ID, name= i
        38: ]
        38: =
        38: ID, name= input
        38: (
        38: )
        38: ;
39:             i = i + 1; }
        39: ID, name= i
        39: =
        39: ID, name= i
        39: +
        39: NUM, val= 1
        39: ;
        39: }
40:     sort(x,0,10);
        40: ID, name= sort
        40: (
        40: ID, name= x
        40: ,
        40: NUM, val= 0
        40: ,
        40: NUM, val= 10
        40: )
        40: ;
41:     i = 0;
        41: ID, name= i
        41: =
        41: NUM, val= 0
        41: ;
42:     while (i < 10)
        42: reserved word: while
        42: (
```

```
        42: ID, name= i
        42: <
        42: NUM, val= 10
        42: )
43:     {           output(x[i]);
        43: {
        43: ID, name= output
        43: (
        43: ID, name= x
        43: [
        43: ID, name= i
        43: ]
        43: )
        43: ;
44:             i = i + 1; }
        44: ID, name= i
        44: =
        44: ID, name= i
        44: +
        44: NUM, val= 1
        44: ;
        44: }
45: }
        45: }
        46: EOF
```