Lexical Analysis(Scanner) Report

- 주하진, 2024062806

Compilation Environment and Method

주어진 `Makefile`을 이용하여 C파일을 컴파일함.
C파일은 `gcc`를 이용해서 컴파일한다.
`Makefile`에서 산출되는 실행파일은 `cminus_cimpl`과 `cminus_lex`가 있으며 각각 `main.c util.c scan.c`, `main.c util.c lex.yy.c`를 컴파일한 오브젝트 파일을 필요로 한다.

`lex.yy.c`는 `flex -o lex.yy.c cminus.l`을 통해 생성된다.

C-Minus Language

C-Minus에서 필요한 토큰타입변수와 그에 대한 설명은 다음과 같다.

특수 토큰

- `ENDFILE` : 파일끝
- `ERROR` : 에러

키워드 토큰

- `IF` : `if`
- `THEN` : `then`
- `ELSE` : `else`
- `WHILE` : `while`
- `RETURN` : `return`
- `INT` : `int`
- `VOID` : `void`

가변길이 토큰

- `ID` : 식별자
- `NUM` : 숫자

기호 토큰

- `ASSIGN` : `=`
- `EQ` : `==`

- `NE` : `!=`
- `LT` : `<`
- `LE` : `<=`
- `GT` : `>`
- `GE` : `>=`
- `PLUS` : `+`
- `MINUS` : `-`
- `TIMES` : `*`
- `OVER` : `/`
- `LPAREN` : `(`
- `RPAREN` : `)`
- `LBRACE` : `[`
- `RBRACE` : `]`
- `LCURLY` : `{`
- `RCURLY` : `}`
- `SEMI` : `;`
- `COMMA` : `,`

토큰에 포함되지 않는 스펙

- `/*` ~ `*/` : 주석 (토큰에 포함하지 않음)

위와 같은 토큰 타입을 기반으로 토크나이징하는 것이 목적이다.

Using `scan.c`

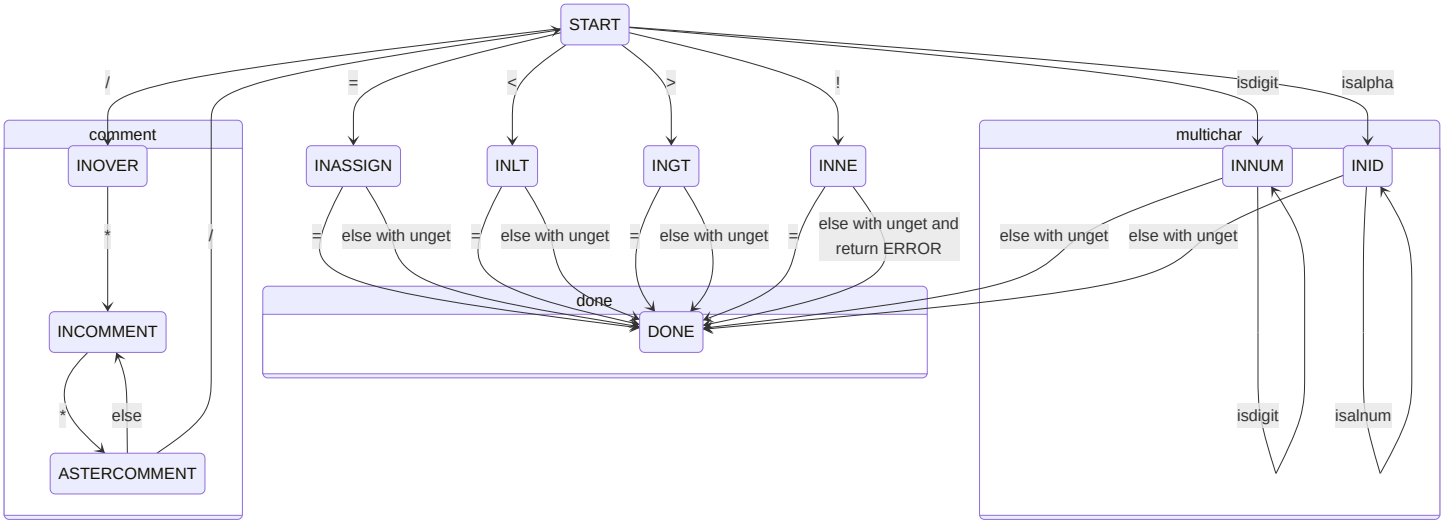
`scan.c`에서는 올바른 `getToken`을 작성해야 한다.

`getToken`을 작성하기에 앞서 전이가능한 `STATE`를 작성한다. 특히 `<`, `>`, `!`, `=`, `/`의 경우에는 단 한 글자만 받는게 아니라 그 다음 문자에 따라 산출할 토큰이 달라질 수 있으므로 그에 따른 `STATE`를 만든다.

결과적으로 필요한 STATE는 다음과 같다.

```
START, INOVER, INCOMMENT, ASTERCOMMENT, INASSIGN, INLT, INGT, INNE, INNUM, INID, DONE
```

이를 이용해 `getToken`의 DFA를 작성할 수 있다.



이를 통해 `scan.c`를 작성하면 된다.

이때 `tokenString`은 항상 넣되 (하지만 NUM, ID 토큰에서만 필요함) comment때만 안 넣으면 된다. `unget`할때도 안넣어야 한다.

Using Lex (cminus.l)

tiny의 lex파일처럼 간단하게 넣고 컴파일하면 된다.

하나 중요한 점은 comment를 구현할 때, `prev`와 `now`를 각 과정에서 계속 업데이트 해가면서 `now == '/' && prev == '*'` 일때까지 계속 `input()`을 받아주면 된다.

Examples & Result

cminus file	result text file

```
/* A program to perform Euclid's
   Algorithm to computer gcd */

int gcd (int u, int v)
{
        if (v == 0) return u;
        else return gcd(v,u-u/v*v);
        /* u-u/v*v == u mod v */
}

void main(void)
{
        int x; int y;
        x = input(); y = input();
        output(gcd(x,y));
}

```

```
C-MINUS COMPILATION: ./test1.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
        6: (
        6: ID, name= v
        6: ==
        6: NUM, val= 0
        6: )
        6: reserved word: return
        6: ID, name= u
        6: ;
        7: reserved word: else
        7: reserved word: return
        7: ID, name= gcd
        7: (
        7: ID, name= v
        7: ,
        7: ID, name= u
        7: -
        7: ID, name= u
        7: /
        7: ID, name= v
        7: *
        7: ID, name= v
        7: )
        7: ;
        9: }
        11: reserved word: void
        11: ID, name= main
        11: (
        11: reserved word: void
        11: )
        12: {
        13: reserved word: int
        13: ID, name= x
        13: ;
        13: reserved word: int
        13: ID, name= y
        13: ;
        14: ID, name= x
        14: =
        14: ID, name= input
        14: (
        14: )
        14: ;
        14: ID, name= y
        14: =
        14: ID, name= input
        14: (
        14: )
        14: ;
        15: ID, name= output
        15: (
        15: ID, name= gcd
        15: (
        15: ID, name= x
        15: ,
        15: ID, name= y
        15: )
        15: )
        15: ;
        16: }
        17: EOF
```

```
1  void main(void)
2  {
3          int i; int x[5];
4
5          i = 0;
6          while( i < 5 )
7          {
8                  x[i] = input();
9
10                 i = i + 1;
11         }
12
13         i = 0;
14         while( i <= 4 )
15         {
16                 if( x[i] != 0 )
17                 {
18                         output(x[i]);
19                 }
20         }
21 }
22
```

```
C-MINUS COMPILATION: ./test2.cm
        1: reserved word: void
        1: ID, name= main
        1: (
        1: reserved word: void
        1: )
        2: {
        3: reserved word: int
        3: ID, name= i
        3: ;
        3: reserved word: int
        3: ID, name= x
        3: [
        3: NUM, val= 5
        3: ]
        3: ;
        5: ID, name= i
        5: =
        5: NUM, val= 0
        5: ;
        6: reserved word: while
        6: (
        6: ID, name= i
        6: <
        6: NUM, val= 5
        6: )
        7: {
        8: ID, name= x
        8: [
        8: ID, name= i
        8: ]
        8: =
        8: ID, name= input
        8: (
        8: )
        8: ;
        10: ID, name= i
        10: =
        10: ID, name= i
        10: +
        10: NUM, val= 1
        10: ;
        11: }
        13: ID, name= i
        13: =
        13: NUM, val= 0
        13: ;
        14: reserved word: while
        14: (
        14: ID, name= i
        14: <=
        14: NUM, val= 4
        14: )
        15: {
        16: reserved word: if
        16: (
        16: ID, name= x
        16: [
        16: ID, name= i
        16: ]
        16: !=
        16: NUM, val= 0
        16: )
        17: {
        18: ID, name= output
        18: (
        18: ID, name= x
        18: [
        18: ID, name= i
        18: ]
        18: )
        18: ;
        19: }
        20: }
        21: }
        22: EOF
```