# Project #1. Scanner

| Created | @Oct 20, 2019 12:57 PM |
| --- | --- |
| Lecture Code | ELE4029 |

## Environment

- Ubuntu 16.04.11 LTS
- gcc (Ubuntu 5.4.0-6ubuntu1~16.04.11) 5.4.0 20160609

# Implementation of C-scanner

간단한 언어인 C-를 스캔하는 scanner를 구현한다.

## Implementation of C-scanner using C-code

### scan.c

두 글자로 이루어진 symbol(`==`, `!=`, `<=`, `>=`, `/*`)의 경우에 하나의 input으로는 인식할 수 없기 때문에, symbol의 첫 번째 글자인 (`=`, `!`, `<`, `>`, `/`)이 들어올 경우 해당 state에 보낸 다음, 다음 input에서 무슨 symbol이 들어왔는지 결정하게 한다.

`INEQ`의 경우 다음 input으로 `=`이 들어오면 해당 symbol이 `EQ`인 것으로 간주하고 `currentToken`을 `EQ`로 정한다. 만약 다른 symbol이면 `ASSIGN`이므로 input을 backup하고, `currentToken`을 `ASSIGN`으로 정한다. `INNE`, `INLT`, `INGT`도 위와 마찬가지로 다음 input을 보고 symbol을 정하면 된다. `INEQ`, `INNE`, `INLT`, `INGT`는 symbol이 정해지면 끝나므로 다음 state는 DONE이 된다.

`INOVER`인 경우에는 다음 input으로 `*`가 들어오면 state가 `INCOMMENT_`로 가야한다. 만약 다른 symbol이면 input을 backup하고, `currentToken`을 `OVER`로 정한다.

`INCOMMENT_` 인 경우에는 주석이므로 save하지 않고 `*`이 input으로 들어올 때까지 계속 받는다. 만약 `*`이 들어오면 다음 input을 미리 읽어서 `/`인지 확인한다. `*/`이라면 주석의 끝이므로 state를 START로 보낸다. 아니라면 미리 읽어온 input을 backup하고 `*/`가 들어올 때까지 계속 읽는다.

## Implementation of C-scanner using lex(flex)

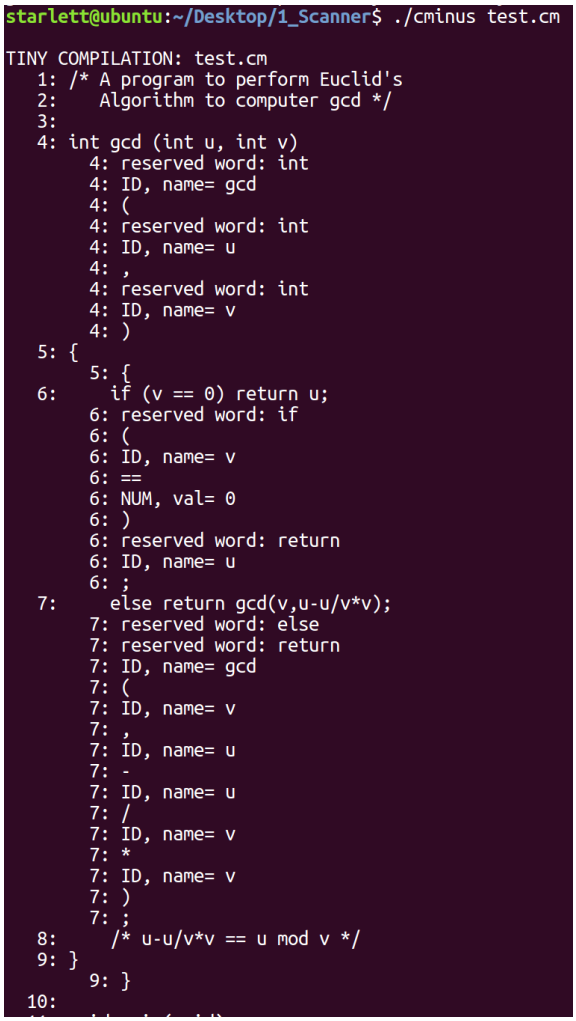
### cminus.l

기존의 `"{"`에서 동작하는 코드를 지우고 주석의 경우 동작할 코드를 추가한다. 주석이 들어올 경우 `lineno`만 증가시키다 `*/`가 들어오면 주석 동작을 끝낸다. `*` 다음에 `/`가 들어왔는지 확인하기 위해 이전 symbol을 담는 history 변수를 만들었다. if문으로 이전에 `*`를 받았고 지금 input이 `/`면 반복문을 탈출해서 동작을 끝내게 했다.

# Result

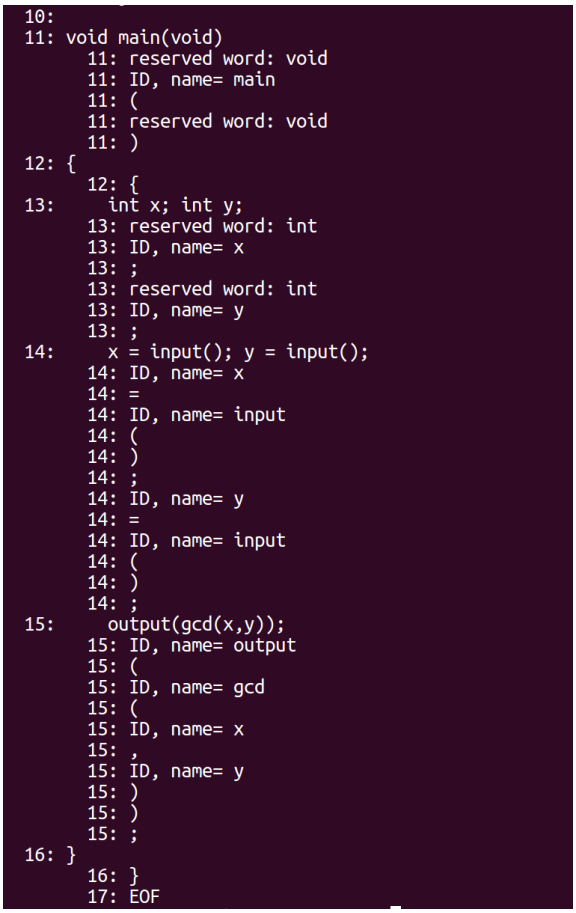
## cminus (using C-code)

### test.cm



```
starlett@ubuntu:~/Desktop/1_Scanner$ ./cminus test.cm

TINY COMPILATION: test.cm
   1: /* A program to perform Euclid's
   2:    Algorithm to computer 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: {
        5: {
   6:     if (v == 0) return u;
        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:     else return gcd(v,u-u/v*v);
        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: ;
   8:     /* u-u/v*v == u mod v */
   9: }
        9: }
  10:
```



```
  10:
  11: void main(void)
        11: reserved word: void
        11: ID, name= main
        11: (
        11: reserved word: void
        11: )
  12: {
        12: {
  13:     int x; int y;
        13: reserved word: int
        13: ID, name= x
        13: ;
        13: reserved word: int
        13: ID, name= y
        13: ;
  14:     x = input(); y = input();
        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:     output(gcd(x,y));
        15: ID, name= output
        15: (
        15: ID, name= gcd
        15: (
        15: ID, name= x
        15: ,
        15: ID, name= y
        15: )
        15: )
        15: ;
  16: }
        16: }
        17: EOF
```

### test2.txt

```
starlett@ubuntu:~/Desktop/1_Scanner$ ./cminus test2.txt

TINY COMPILATION: test2.txt
   1: void main(void)
        1: reserved word: void
        1: ID, name= main
        1: (
        1: reserved word: void
        1: )
   2: {
        2: {
   3:     int i; int x[5];
        3: reserved word: int
        3: ID, name= i
        3: ;
        3: reserved word: int
        3: ID, name= x
        3: [
        3: NUM, val= 5
        3: ]
        3: ;
   4:
   5:     i = 0;
        5: ID, name= i
        5: =
        5: NUM, val= 0
        5: ;
   6:     while( i < 5 )
        6: reserved word: while
        6: (
        6: ID, name= i
        6: <
        6: NUM, val= 5
        6: )
   7:     {
        7: {
   8:         x[i] = input();
        8: ID, name= x
        8: [
        8: ID, name= i
        8: ]
        8: =
        8: ID, name= input
        8: (
        8: )
        8: ;
   9:
```

```
  10:         i = i + 1;
       10: ID, name= i
       10: =
       10: ID, name= i
       10: +
       10: NUM, val= 1
       10: ;
  11:     }
       11: }
  12:
  13:     i = 0;
       13: ID, name= i
       13: =
       13: NUM, val= 0
       13: ;
  14:     while ( i <= 4 )
       14: reserved word: while
       14: (
       14: ID, name= i
       14: <=
       14: NUM, val= 4
       14: )
  15:     {
       15: {
  16:         if ( x[i] != 0 )
       16: reserved word: if
       16: (
       16: ID, name= x
       16: [
       16: ID, name= i
       16: ]
       16: !=
       16: NUM, val= 0
       16: )
  17:         {
       17: {
  18:             output(x[i]);
       18: ID, name= output
       18: (
       18: ID, name= x
       18: [
       18: ID, name= i
       18: ]
       18: )
       18: ;
  19:         }
       19: }
  20:     }
       20: }
  21: }
       21: }
       22: EOF
```

# cminus_flex (using flex)

## test.cm

```
starlett@ubuntu:~/Desktop/1_Scanner$ ./cminus_flex 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
        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
```

## test2.txt

```
starlett@ubuntu:~/Desktop/1_Scanner$ ./cminus_flex test2.txt

TINY COMPILATION: test2.txt
        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
```