컴파일러 설계

Project #1 : Scanner

컴퓨터소프트웨어학부
2018008395
박정호

1. 실행 환경

우분투에서 진행했고, 버전은 16.04.12이다. GCC 버전은 다음과 같다.

```
pch@pch-VirtualBox:~$ gcc --version
gcc (Ubuntu 5.4.0-6ubuntu1~16.04.12) 5.4.0 20160609
Copyright (C) 2015 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
```

2. Implementation

A. Implementation of C-Scanner using C-code

NUM이나 ID, 혹은 기타 한 글자 Token의 처리는 기존 코드와 같다. 여기서는 C-에서 추가된 사항에 대해서만 다뤄보도록 하겠다.

i. Multi-Line Comment의 처리

주석의 시작은 "/*"로 이루어진다. 그러나, '/'는 나눗셈 연산자이기에, input 한 글자로는 어느 Token인지 알 수 없었다. 따라서 '/'를 받았을 때 일단 state를 INOVER로 두고 그 다음 문자를 받았다.

여기서 그 다음 문자가 '*'라면 이 다음부터는 주석이 계속된다고 보고, state를 INCOMMENT로 바꾸었고, 아니라면 나눗셈 연산자이므로 읽어온 문자를 unget한 다음, currentToken을 OVER로 두었다.

state가 INCOMMENT일 경우, 입력받는 모든 문자를 token string에 저장하지 않고 무시하는데, 단, 문자가 '*'인 경우 그다음 문자가 '/'인지 확인한다. 만약 맞다면 주석의 종료를 의미하므로 state를 START로 바꾸었고, 아니라면 unget해서 다시 위의 과정을 반복했다. 물론 여기서 중간에 EOF를 만날 경우, ENDFILE를 내놓는 처리는 제대로 해주었다.

ii. "<=", ">=", "==", "!=" 등 두 글자 Token의 처리

네 가지 경우 모두 처리 방법이 같으므로, "=="를 예시로 진행하겠다. START state에서 '='를 입력받은 경우, 이 Token이 "=="일지 "="일지 알 수 없으므로, state를 INEQ로 두었다. 이후, INEQ state에서는 현재 문자가 '='일 경우 currentToken을 EQ로 두었고, 이외의 경우에는, 문자를 unget한 다음 currentToken을 ASSIGN으로 두었다.

두 글자 Token은 모두 이와 같은 처리를 거친다. 예외로, "!="의 경우 '!' 다음에 '='가 오지 않을 경우 ERROR로 본다. "!"라는 Token 정의가 존재하지 않기 때문이다.

B. Implementation of C-Scanner using lex(flex)

여타 구조는 단순히 Token 자체를 스펙으로 넣으면 되기 때문에 특별한 구현은 없다. 단, 주석의 구현에서 약간의 변동이 있으므로, 이에 대한 설명만 진행하겠다.

i. Multi-Line Comment의 처리

기존의 Tiny는 주석을 "{", "}" 로 나타냈었다. 이는 양 끝이 한 글자 Token이기에 처리하기 간단했는데, C-는 "/*", "*/"라는 두 글자 Token을 사용하기에 조금 처리가 까다로웠다. 우선 문자를 저장할 변수 c1, c2를 두고, c2를 현재 입력 받은 문자, c1을 그 이전 문자로 보고 구현을 진행했다.

이후는 기존의 주석 구현과 거의 같다. 단지 주석으로 보는 범위의 조건이 c1이 '*' 이고, c2가 '/'가 될 때까지라는 점이 다른데, 이는 주석의 종료를 나타내는 Token이 "*/"이기 때문이다.

3. 실행 결과

실행에 사용한 input은 Blackboard에 제공된 test1.txt, test2.txt를 사용했음을 밝힌다.

A. scanner_cimpl,

with test1.txt (test1.cm)

```
pch@pch-VirtualBox:~/2020_ELE4029_2018008395/1_Scanner$ ./scanner_cimpl test1.cm

C-MINUS COMPILATION: test1.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:
  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
```

B. scanner_cimpl,

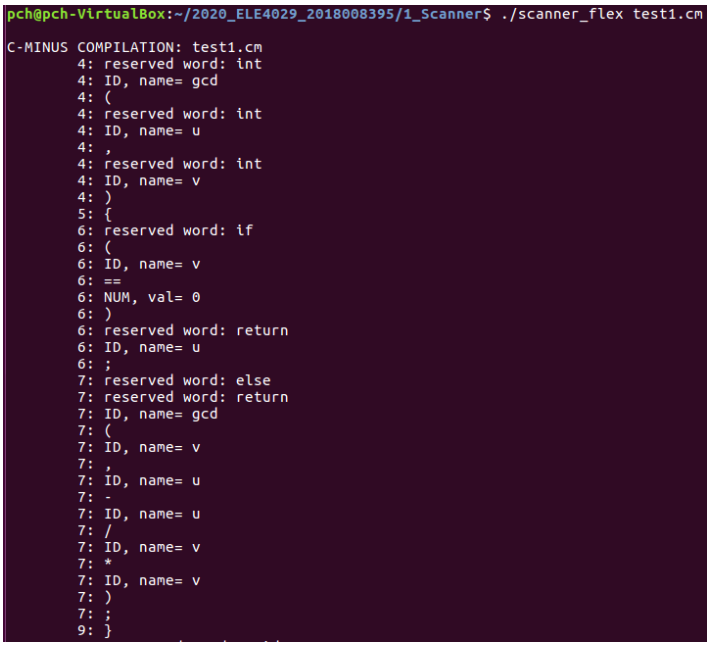
with test2.txt (test2.cm)

```
pch@pch-VirtualBox:~/2020_ELE4029_2018008395/1_Scanner$ ./scanner_cimpl test2.cm

C-MINUS COMPILATION: test2.cm
   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:
```

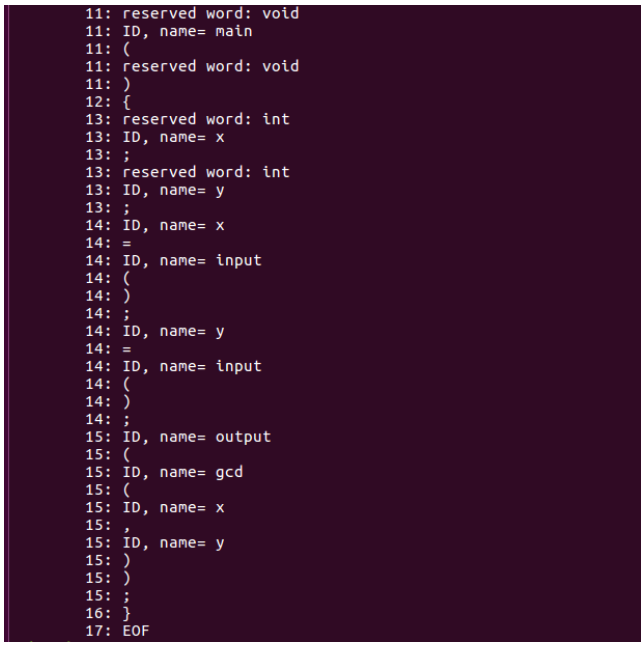
```
  10:         i = i + 1;
       10: ID, name= i
       10: =
       10: ID, name= i
       10: +
       10: NUM, val= 1
       10: ;
  11:     }
       11: }
  12:
  13:     i = 0;
       13: ID, name= i
       13: =
       13: NUM, val= 0
       13: ;
  14:     while( i <= 4 )
       14: reserved 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
```

C. scanner_flex

with test1.txt(test1.cm)



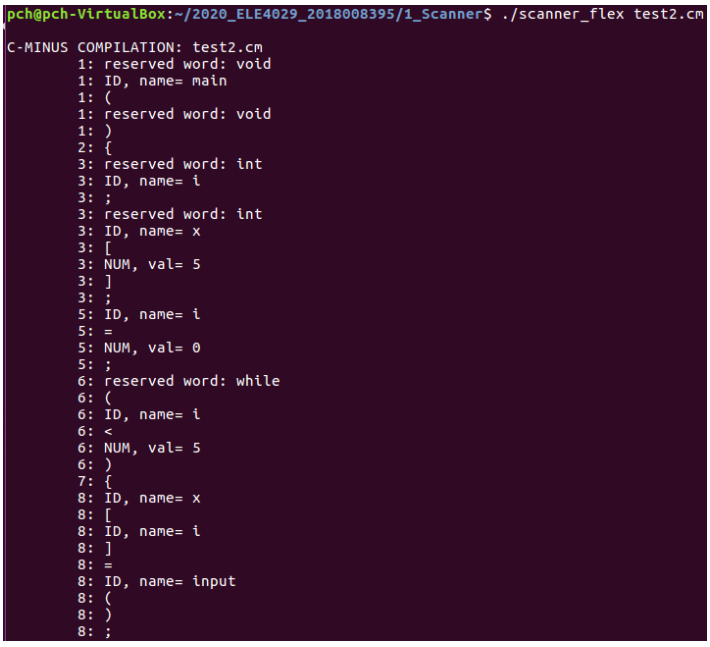
```
pch@pch-VirtualBox:~/2020_ELE4029_2018008395/1_Scanner$ ./scanner_flex test1.cm

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: }
```



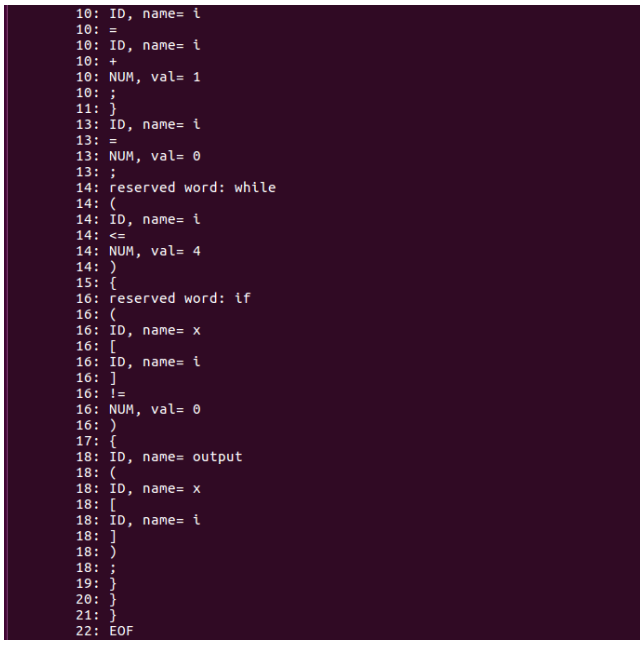
```
       11: reserved word: void
       11: ID, name= main
       11: (
       11: reserved word: void
       11: )
       12: {
       13: reserved word: int
       13: ID, name= x
       13: ;
       13: reserved word: int
       13: ID, name= y
       13: ;
       14: ID, name= x
       14: =
       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
```

D. scanner_flex

with test2.txt(test2.cm)



```
pch@pch-VirtualBox:~/2020_ELE4029_2018008395/1_Scanner$ ./scanner_flex test2.cm

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
```