컴파일러설계 Project 1(Scanner)

2016026599
정보시스템학과 황성우

1. Compilation method and environment (컴파일 방법과 개발환경)

컴파일과 실습은 모두 Virtualbox 안에서 Ubuntu 14.04 으로 진행하였습니다. 컴파일은 gcc 5.4.0을 사용하였고 lex는 apt-get 을 이용하여 lex 2.6.0을 사용했습니다. Makefile은 포탈에 올려주신것을 그대로 사용하였습니다.

2. Explanation about how to implement and how to operate

수정하여 구현한 파일은 크게 globals.h, main.c , scan.c , util.c, cminus.l로 5가지입니다. (Makefile은 수정하지 않았습니다.)

1) globals.h
   기본 globals.h에서 과제 안내 pdf의 hint를 기반으로 수정하였습니다. 원래 파일에는 있었지만 이번 프로젝트에서는 사용되지 않는 토큰들은 /* discarded */ 주석을 사용해 분리만 해놓고 삭제하지는 않았습니다
2) main.c
   NO_PARSE 부분과 EchoSource,Tracescan부분만 TRUE로 설정해주었습니다
3) scan.c
   1번처럼 과제 안내 pdf 안의 hint에 나와있는 새로운 토큰들을 enum인 statetype에 추가해주었습니다. 기존의 토큰중 INASSIGN을 제외한 토큰들은 삭제하지 않았습니다. reservedwords 구조체 역시 과제에 있는 hint 처럼 추가해주었습니다. 그 다음 getToken 함수를 수정해주었습니다. 이전의 수정점은 대부분이 hint만 보고 추가해주는거라 어렵지 않았는데, getToken 함수의 수정을 위해서는 코드 전체에 대한 이해가 필수적이었습니다.
   1개 문자만을 사용하는 토큰('{','}')은 쉽게 추가할수 있었지만 나머지는 일단 assign 연산자의 형태를 보고 배워서 추가하는 식으로 수정해주었습니다. (<=,>=,==,!=)등을 수정해주었습니다. getToken 함수에서 가장 헷갈리는 부분은 주석처리 였습니다. 일단 INCOMMENT와 INCOMMENT_의 차이를 먼저 알아야 했습니다. gettoken의 역할이 dfa와 비슷하다는것이 생각나서 아주 간단하게 dfa를 그려서 생각해보니 해결할 수 있었습니다. /를 만났을때와 그다음은 *를 만났을때, 그 후에 *를 만나기 전까지는 모두 주석으로 받아들이고 마지막 /를 만났을때 주석을 마치는 구성을 구현했습니다.
4) util.c
   대부분은 pdf안에 나와있는 hint대로 처리했는데, 그 후에 테스트를 해보니 reservedword에 추가한 새로운 토큰들이 (void,int) error로 표기되어서 의아했습니다. 그 후에 기존의 reservedword(if,else)등이 어떻게 처리되는지 이해하고 추가해주어서 수정하였습니다. 기존의 토큰들은 주석처리 해주었습니다.
5) cmius.l

기존의 문법이 직관적이어서 작성하는데 어렵지 않았지만 역시 주석부분이 가장 어려웠습니다.

기존의 토큰들은 assign을 보고 이전의 scan.c 처럼 dfa를 고려하는것이 아니라는것을 알게 되어서 쉽게 추가해주었습니다. 주석부분은 '/*'로 체크하게 하고 직전 문자를 체크하는 prev 변수를 통하여 /* */ 검출하는 기능을 구현했습니다.

6) How to operate?

makefile이 이미 구성되어 있기 때문에 make clean을 먼저 한 후 make를 하면 scanner_cimpl 파일과 scanner_flex 파일이 생성됩니다 . 그 후에 ./scanner_cimpl <target cminus file> 혹은 ./scanner_flex <target cminus file>을 입력하면 아래와 같은 식의 결과를 확인 할 수 있습니다.

3. Example and Result Screenshot

1)scan.c 를 이용해 만든 scanner_cimpl 실행 결과

```
ubuntu@machine ~/Desktop/2019_ELE4029_2016026599/1_Scanner
 % ./scanner_cimpl gcd_test.cm

TINY COMPILATION: gcd_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= 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: ID, name= gcd
        15: (
        15: ID, name= x
        15: ,
        15: ID, name= y
        15: )
        15: )
        15: ;
 16: }
        16: }
        17: EOF
ubuntu@machine ~/Desktop/2019_ELE4029_2016026599/1_Scanner
% $
```

2.flex를 사용하여 만들어진 cminus.l을 수정하여 만든 scanner_flex 실행결과

```
ubuntu@machine ~/Desktop/2019_ELE4029_2016026599/1_Scanner
% ./scanner_flex gcd_test.cm

TINY COMPILATION: gcd_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
```