scanner_report

태그

Compilation

- compilation environment

  `Ubuntu 20.04.6 LTS` , `gcc 9.4.0`

1. method #1. C implementation

   `make cminus_cimpl`

   `./cminus_cimpl ./TestCase/test.1.txt`

2. method#2. Lex implementation

   `sudo apt-get install flex`

   `make cminus_lex`

   `./cminus_lex ./TestCase/test.1.txt`

Implementation

C implementation

- **globals.h** 수정

과제 명세에 맞게 reserved words 와 special symbols 를 수정해 주었다.

- scan.c 수정

해당 소스코드에서는 getToken() 함수를 수정해주었다.

한 글자만으로 결정이 가능한 symbol은 공통적으로 하나의 switch 문에 case 를 나누어서 symbol 이 들어온 경우 해당 토큰으로 currentToken 을 변경시켜주었다.

추가로 `>= <= != ==` 등과 같이 두개의 글자를 인식해야 결정할 수 있는 symbol은 위와 따로 분리하여서 앞의 symbol 을 인식했을 때, 그 다음 글자를 추가로 확인하는 코드를 작성하여 스캐너를 구현하였다. 이를 위해 state 들을 추가해주어야 했다.

symbol 이 정해졌을 경우 currentToken 을 해당 symbol로 설정해주었고, 다음 글자를 읽었을 때 그 전 string 까지가 symbol 에 해당하는 경우 위와 동일하게 currentToken 을 해

당 symbol 로 설정한 뒤, ungetNextChar() 을 사용하여 한 글자 전으로 되돌려 놓았다.

주석같은 경우, "/" 를 읽었을 경우 `INOVER` state 로 들어오고, 해당 state 에서 "*" 을 읽으면 `INCOMMENT` state 로 들어온다. 해당 state 에서 계속해서 문자를 읽고 만약 EOF 이면 currentToken 을 ENDFILE 로 설정해주고 state 를 DONE 으로 처리해준다. 또는 "*" 를 읽으면 `INCOMMENT_`state 로 변경해주어서 주석의 닫는 부분 처리를 해주었다. INCOMMENT_ state 에서도 마찬가지로 EOF 라면 currentToken 을 ENDFILE 로 변경해주고 state 를 DONE 으로 처리해준다. 또는 "/" 를 만나면 주석이 끝나고 그 다음 글자를 새롭게 읽을 수 있도록 state를 START로 설정해주었다.

또한 ID = letter(letter|digit)* 이 될 수 있도록 수정해주었다.

- util.c

마지막으로 util.c 파일에서는 이 전에 지정해 주었던 토큰들을 출력해주는 코드들을 추가하여 최정적으로 명세에 해당하는 출력이 나올 수 있도록 해주었다.

LEX

cminus.l 파일에서 주어진 토큰들을 추가하여 return 값을 설정해주었고, ID = letter(letter|digit)* 이 될 수 있도록 수정해주었다.

주석 부분 은 /가 들어온 경우 무한루프를 추가하였으며, 문자를 하나씩 읽어올 수 있도록 input() 을 이용 하였다. 이때 개행이 된 경우 줄 번호를 추가해 주기 위해서 lineno++ 를 사용하여 줄 번호를 알 맞게 증가시켜주었고, / 이후 아무 내용이 들어오지 않은 경우, break 를 통해 루프를 종료시켜주 었다. 이후 *가 들어오면 그 이후의 글자를 인식하고 마지막으로 / 가 나온 경우, 주석이 종료되 었으므로 루프를 종료시켜 주었다.

Test

TestCase > test.1.txt

```
/* 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));
}

```

```
jieunpark@ubuntu ~/compiler/loucomp (0.144s)
./cminus_cimpl ./TestCase/test.1.txt

C-MINUS COMPILATION: ./TestCase/test.1.txt
        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
```

```
jieunpark@ubuntu ~/compiler/loucomp (0.144s)
./cminus_lex ./TestCase/test.1.txt

C-MINUS COMPILATION: ./TestCase/test.1.txt
        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
```

TestCase > test.2.txt

```
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
```

```
jleunpark@ubuntu ~/compiler/loucomp (0.147s)
./cminus_cimpl ./TestCase/test.2.txt

C-MINUS COMPILATION: ./TestCase/test.2.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
```

```
jleunpark@ubuntu ~/compiler/loucomp (0.099s)
./cminus_lex ./TestCase/test.2.txt

C-MINUS COMPILATION: ./TestCase/test.2.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
```