컴파일러설계

프로젝트 보고서

#1 Scanner

Hanyang Univ. CSE

2014004411 김시완

1. 개발환경

1) Linux Kernel 버전 확인

```
siwankim@siwankim-VirtualBox:~$ cat /proc/version
Linux version 4.15.0-36-generic (buildd@lcy01-amd64-017) (gcc version 5.4.0 2016
0609 (Ubuntu 5.4.0-6ubuntu1~16.04.10)) #39~16.04.1-Ubuntu SMP Tue Sep 25 08:59:2
3 UTC 2018
```

2) Gcc 버전 확인

```
siwankim@siwankim-VirtualBox:~$ gcc --version
gcc (Ubuntu 5.4.0-6ubuntu1~16.04.10) 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. 컴파일 방법

아래와 같이 make all 명령어를 통해 컴파일을 진행할 수 있습니다.

```
siwankim@siwankim-VirtualBox:~/2018_ELE4029_2014004411/assignment1$ make all
gcc  -c main.c
main.c:48:1: warning: return type defaults to ‘int’ [-Wimplicit-int]
 main( int argc, char * argv[] )
 ^
gcc  -c util.c
util.c:124:8: warning: type defaults to ‘int’ in declaration of ‘indentno’ [-Wim
plicit-int]
 static indentno = 0;
```

3. 코드 설명

1) 직접 tiny 파일을 수정하는 방법

A. 기존 코드 수정

```
        state = INID;
      // else if (c == ':')
      //   state = INASSIGN;
      else if ((c == ' ') || (c == '\t') || (c == '\n'))
        save = FALSE;
      // else if (c == '{')
      // { save = FALSE;
      //   state = INCOMMENT;
      // }
```

```
// case INCOMMENT:
//   save = FALSE;
//   if (c == EOF)
//   { state = DONE;
//     currentToken = ENDFILE;
//   }
//   else if (c == '}') state = START;
//   break;
// case INASSIGN:
//   state = DONE;
//   if (c == '=')
//     currentToken = ASSIGN;
//   else
//   { //backup in the input
//     ungetNextChar();
//     save = FALSE;
//     currentToken = ERROR;
//   }
//   break;
```

기존 ':=' symbol 을 통한 대입 연산자와 '{' symbol 을 통한 주석 연산자는 더 이상 사용하지 않으므로 위와 같이 코드를 삭제 해주었습니다.

```
// case '=':
//   currentToken = EQ;
//   break;
// case '<':
//   currentToken = LT;
//   break;
case '+':
  currentToken = PLUS;
  break;
case '-':
  currentToken = MINUS;
  break;
case '*':
  currentToken = TIMES;
  break;
// case '/':
//   currentToken = OVER;
//   break;
```

또한, '='로 시작하는 keyword 가 '='와 '=='로, '<'로 시작하는 keyword 가 '<'와 '<='로, '/'로 시작하는 keyword 가 '/', '/*'로 증가하였으므로 기존 코드를 수정해 주었습니다.

B. Keyword '='와 '=='의 구분

```
      else if (c == '=')
      { save = FALSE;
        state = INEQ;
      }
```

STATE 'START'에서 문자열 '='를 입력 받았을 때 STATE 값을 INEQ 로 변경시켜 주었습니다.

다음 INEQ 상태에서

```
case INEQ:
 state = DONE;
 if ( c == '=')
  currentToken = EQ;
 else
 { currentToken = ASSIGN;
  ungetNextChar();
 }
 break;
```

와 같이 다음 문자열 '='를 입력 받으면 keyword '=='(EQ)을 입력 받은 것이므로 currentToken 을 EQ 으로 지정해주고, STATE 는 DONE 으로 변경시켜 주었습니다.

이 외에, 다른 문자열을 입력 받은 경우 keyword '='(ASSIGN)을 입력 받은 후 하나의 문자열을 더 읽은 상태이므로 currentToken 은 ASSIGN 으로 지정해 준 다음, ungetNextChar() 함수를 이용하여, 문자열을 하나 덜 읽은 이전 상태로 다시 변경시켜 주었습니다. 마찬가지로 STATE 는 DONE 으로 상태를 변경시켜 주었습니다.

C. Keyword '<'와 '<='의 구분, '>'와 '>='의 구분

위와 마찬가지로 문자열 '<'을 입력 받았을 때는 STATE 를 INLT 로 변경해준 다음, 다음 문자열을 읽었을 때 문자열 '='을 읽었을 때는 currentToken 을 LE 로, 다른 문자열을 읽었을 때는 currentToken 을 LT 로 지정해 준 다음, STATE 를 DONE 으로 변경시켜 주었습니다.

문자열 '>'을 입력 받았을 때는 STATE 를 INGT 로 변경해 준 다음, 다음 문자열을 읽었을 때 '='인 경우 currentToken 을 GE 로, 다른 문자열을 읽었을 때 GT 로 지정해 주었습니다.

D. Keyword '/'와 '/*'의 구분

```
 else if (c == '/')
 { save = FALSE;
  state = INOVER;
 }
```

문자열 '/'을 입력 받았을 때 state 를 INOVER 로 변경시켜준 다음, 다음 문자열을 입력 받아주도록 하였습니다.

```
case INOVER:
 if ( c == '*')
 { state = INCOMMENT_;
  save = FALSE;
 }
 else
 { state = DONE;
  currentToken = OVER;
  ungetNextChar();
 }
 break;
```

다음 문자열을 입력 받았을 때 '*'인 경우 keyword '/*'를 입력받은 경우이므로 state 를 'INCOMMENT_'로 변경 시켜준 다음, 마찬가지로 다음 문자열을 입력 받아주었고 아닌 경우 Keyword '/'를 입력 받은 경우이므로 ungetNextChar 함수를 호출한 다음 currentToken 을 OVER 로 지정 해주었다.

```
case INCOMMENT_:
 save = FALSE;
 if(c == EOF)
 { state = DONE;
  currentToken = ENDFILE;
 }
 else if(c == '*')
 { c = getNextChar();
  if(c == '/')
   state = START;
  else if(c == EOF)
  { state = DONE;
   currentToken = ENDFILE;
  }
  else
   ungetNextChar();
 }
 break;
```

STATE 가 INCOMMENT_인 경우 주석의 끝을 알리는 '*/' 문자열을 입력 받거나, 'EOF'을 입력 받을 때까지 INCOMMENT_ 상태가 계속 지속되게끔 해주었다. 중간에 'EOF'를 입력(파일이 끝날 경우)받을 경우 STATE 는 DONE 으로 변경해주었으며 currentToken 은 ENDFILE 로 지정해주었다. 또한, 주석이 종료될 경우 STATE 를 START 로 변경해주어 다시 문자열을 인식할 수 있게끔 변경해주었으며, 주석 내에서 받는 문자열은 저장할 필요가 없으므로 SAVE 를 FALSE 로 지정해주었다.

2) Flex 파일을 이용하는 방법

```
"/*"          { char c = '0';
              while(1)
              { if( c != '*') c = input();
               if (c == EOF) break;
               if (c == '*')
               { c = input();
                if((c == '/') || (c == EOF)) break;
               }
               if (c == '₩n') lineno++;
              }
             }
.             {return ERROR;}
```

다른 문자열을 입력 받는 경우는 기존 경우들과 비슷하게 구현을 해주었으며, 기존 '{' keyword 를 통해 주석을 처리해주는 경우를 위와 같이 변경해 주었다. '/*' 문자열을 인식해 주석이 시작되는 경우 '*/'를 입력 받거나 파일이 종료되는 경우에만 while 문을 빠져나와 주석이 종료될 수 있으며 중간에 개행 문자열이 입력되는 경우만 줄 수를 출력하기 위해 lineno 를 증가시켜 주었습니다.

```
"="           {return ASSIGN;}
"=="          {return EQ;}
```

기존 ASSIGN keyword 인 ':='를 '='로 변경시켜주었습니다.

4. 결과

```
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:
```

```
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
```

과제 ppt 에 존재하는 예제 결과(tiny.l 이용)

```
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
```

과제 ppt 에 존재하는 예제 결과(cminus_flex.l 이용)