Project 1: Scanner

2016024875 손정우

1. Compilation methods and environment

To compile the project, I used Makefile example given by the TA. The environment that I build this project was as below:

- Ubuntu 16.04.7 (64-bits, on VMware Workstation)
- gcc 5.4.0 20160609
- flex 2.6.0

2. Implementation of C-scanner

1) Using C-code

In this implementation, I focused on modifying globals.h, main.c, util.c, and scan.c as project specification suggested. (Note that Tiny's reserved words were left unchanged as specification hints did. I will remove these words in future projects.) Among these source codes, scan.c was the most important as it has getToken() function.

Processing the comment section was the trickiest part since we need to check whether it is just '/' or '/*'. Also, note that any characters can exist between the '/*' and '*/', so we need to consider the transition between the INCOMMENT_ state and INCOMMENT state. The result of solving these problems is as the following code.

```
/* scan.c */

TokenType getToken(void)
{  /* 중략 */

   while (state != DONE)
   { int c = getNextChar();
     save = TRUE;
     switch (state)
     { case START:
         if (isdigit(c))
           state = INNUM;
         else if (isalpha(c))
           state = INID;
         else if ((c == ' ') || (c == '\t') || (c == '\n'))
           save = FALSE;
         else if (c == '/')
         { save = FALSE;
           state = INOVER;
         }
         else if (c == '=')
           state = INEQ;
         else if (c == '!')
           state = INNE;
```

```
        else if (c == '>')
          state = INLT;
        else if (c == '<')
          state = INGT;
        else
        { state = DONE;
          switch (c)
          {
            /* 중략 */

          }
        }
        break;
      case INOVER:
        save = FALSE;
        if (c == '*')
          state = INCOMMENT_;
        else
        { save = TRUE;
          state = DONE;
          ungetNextChar();
          currentToken = OVER;
        }
        break;
      case INCOMMENT_:
        save = FALSE;
        if (c == '*')
          state = INCOMMENT;
        else
          state = INCOMMENT_;
        break;
      case INCOMMENT:
        save = FALSE;
        if (c == EOF)
        { state = DONE;
          currentToken = ENDFILE;
        }
        else if (c == '/')
          state = START;
        else if (c == '*')
          state = INCOMMENT;
        else
          state = INCOMMENT_;
        break;
      case INEQ:
        save = FALSE;
        state = DONE;
        if (c == '=')
          currentToken = EQ;
        else
        { /* backup in the input */
          ungetNextChar();
          save = TRUE;
```

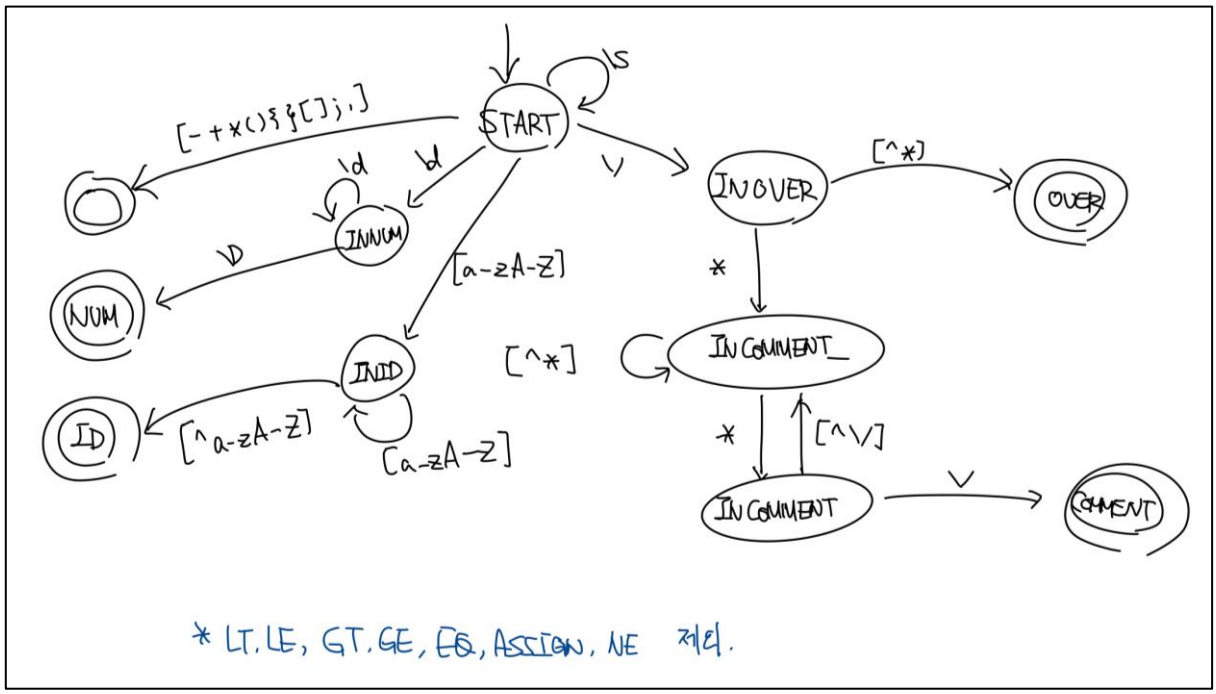
```
            currentToken = ASSIGN;
          }
          break;
        /* 중략 */

      }
      /* 중략 */

} /* end getToken */
```

2) Using flex by Tiny.l

To change Tiny.l to cminus.l, I first approached by drawing an automata that processes cminus like below.



Based on this automata, I implemented cminus.l like below.

```
/* cminus.l */

/* 중략 */

%%

"if"            {return IF;}
"else"          {return ELSE;}
"while"         {return WHILE;}
"return"        {return RETURN;}
"int"           {return INT;}
"void"          {return VOID;}
"then"          {return THEN;}   /* discarded */
"end"           {return END;}    /* discarded */
"repeat"        {return REPEAT;} /* discarded */
"until"         {return UNTIL;}  /* discarded */
"read"          {return READ;}   /* discarded */
"write"         {return WRITE;}  /* discarded */
"="             {return ASSIGN;}
```

```
"=="             {return EQ;}
"!="             {return NE;}
"<"              {return LT;}
"<="             {return LE;}
">"              {return GT;}
">="             {return GE;}
"+"              {return PLUS;}
"-"              {return MINUS;}
"*"              {return TIMES;}
"/"              {return OVER;}
"("              {return LPAREN;}
")"              {return RPAREN;}
"["              {return LBRACE;}
"]"              {return RBRACE;}
"{"              {return LCURLY;}
"}"              {return RCURLY;}
";"              {return SEMI;}
","              {return COMMA;}
{number}         {return NUM;}
{identifier}     {return ID;}
{newline}        {lineno++;}
{whitespace}     {/* skip whitespace */}
"/*"             { char b, c;
                   b = 0;
                   do
                   { c = input();
                     if (c == EOF) break;
                     else if (c == '\n') lineno++;
                     else if (b == '*' && c == '/') break;
                     b = c;
                   } while (TRUE);
                 }
.                {return ERROR;}

%%

/* 후략 */
```

3. Example Results

By referring to the provided example file and the result value, I confirmed that the codes I implemented work normally. The following are screen shots showing that the code works ordinarily.

<table>
<tr><th>Testcase</th><th>Using C codes</th><th>Using flex</th></tr>
<tr><td>test.1.txt</td><td>

```
jw@ubuntu:~/2020_ELE4029_2016024875/1_Scanner
jw@ubuntu ~/2020_ELE4029_2016024875/1_Scanner master ± ./scanner_cimpl ./testcase/test.1.txt

C-MINUS COMPILATION: ./testcase/test.1.txt
   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: (
```

</td><td>

```
jw@ubuntu:~/2020_ELE4029_2016024875/1_Scanner
jw@ubuntu ~/2020_ELE4029_2016024875/1_Scanner master ± ./scanner_flex ./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: =
```

</td></tr>
<tr><td>test.2.txt</td><td>

```
jw@ubuntu:~/2020_ELE4029_2016024875/1_Scanner
jw@ubuntu ~/2020_ELE4029_2016024875/1_Scanner master ± ./scanner_cimpl ./testcase/test.2.txt

C-MINUS COMPILATION: ./testcase/test.2.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: +
```

</td><td>

```
jw@ubuntu:~/2020_ELE4029_2016024875/1_Scanner
jw@ubuntu ~/2020_ELE4029_2016024875/1_Scanner master ± ./scanner_flex ./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
```

</td></tr>
<tr><td colspan="3">*Note that some of the results have been omitted to meet the report specifications. (less than 5 pages)</td></tr>
</table>