28/35

```
1  //CalcLex.h
2  //Charlie Ang
3  //November 13, 2016
4  //CSC 3310 Autumn 2016
5  //Header file including interface specifications for CalcLex
6
7  #ifndef _CALCLEX_H
8  #define _CALCLEX_H
9
10 #include <iostream>
11 #include <fstream>
12 #include <cstring>
13 #include <string>
14 #include <iomanip>
15 using namespace std;
16
17 enum CalcGrammarTokens{
18     EOFSY = 0, LPAREN, RPAREN, ADDOP, SUBOP, MULTOP, DIVOP, ASSIGNOP, ID,
       NUMCONST, READSY, WRITESY
19 };
20
21 #define YYTEXT_MAX 100
22 extern char yytext[YYTEXT_MAX]; //Global token text buffer
23
24 bool yylexopen(const char filename[]);
25 void yytextclear();
26 void yytextappend();
27 int yylex();
28
29 bool followingCharsMatch(int c, string target);
30 string tokenType(int token);
31 //int yyparse();
32
33 #endif
```

```
1  //CalcLexMain.cpp
2  //Charlie Ang
3  //November 13, 2016
4  //CSC 3310 Autumn 2016
5  //CalcLex Main program
6
7  #include "CalcLex.h" //implementation file includes header file
8
9  int main(int argc, char *argv[])
10 {
11     //Pick up commandline input filename, if any
12     if (argc > 1 && (!yylexopen(argv[1])))
13     {
14         cout << "Error: Cannot open input file " << argv[1] << endl;
15         exit(1);
16     }
17
18     int token;
19     string tokenTypeSym;
20     int numTokens = 0;  //keeps track of number of tokens
21     while ((token = yylex()) != EOFSY)      //reads through file token by token
         until end of file reached
22     {
23         tokenTypeSym = tokenType(token);
24         cout << "tok = " << setw(2) << setfill('0') << token << " " <<
             tokenTypeSym << " " << "(" << yytext << ")" << endl; //tok = 10 READSY
             (read)
25         numTokens++;
26     }
27
28     //at this point, token should be EOF
29     if (token == EOFSY)
30     {
31         tokenTypeSym = tokenType(token);
32         cout << "tok = " << setw(2) << setfill('0') << token << " " <<
             tokenTypeSym << " " << "(" << yytext << ")" << endl; //tok = 00 EOFS-$$
33     }
34
35     cout << endl;
36     cout << "Number of tokens = " << numTokens << endl; //Number of tokens = 15
37
38     return 0;
39 }
40
41 //Identifies what the token string value is and returns the string
42 string tokenType(int token)
43 {
44     switch (token)
45     {
46         case 0:
47             return "EOFSY--$$"; //end of file symbol
48         case 1:
```

```
49              return "LPAREN";    //left parenthesis
50          case 2:
51              return "RPAREN";    //right parenthesis
52          case 3:
53              return "ADDOP"; //add operation
54          case 4:
55              return "SUBOP"; //subtract operator
56          case 5:
57              return "MULTOP";    //multiply operator
58          case 6:
59              return "DIVOP"; //divide operator
60          case 7:
61              return "ASSIGNOP";  //assignment operator
62          case 8:
63              return "ID";    //ID
64          case 9:
65              return "NUMCONST";  //numeric constant
66          case 10:
67              return "READSY";    //read symbol
68          case 11:
69              return "WRITESY";   //write symbol
70          default:
71              return "Not a valid token for Calc Grammar";
72      }
73  }
74
```

```
 1 //CalcLex.cpp
 2 //Charlie Ang
 3 //November 13, 2016
 4 //CSC 3310 Autumn 2016
 5 //Implementaiton file for CalcLex.h
 6
 7 #include "CalcLex.h" //implementation file includes header file
 8
 9 char yytext[YYTEXT_MAX];    //Global token text buffer
10 static int yytextpos = 0;   //allocated statically to extend across the entire
     program
11
12 ifstream fin;
13 bool yylexopen(const char filename[]){
14     fin.open(filename, ios_base::in);
15     return fin.is_open();   //true if file opened, false otherwise
16 }
17
18 //resets token buffer
19 void yytextclear(){
20     yytextpos = 0;
21     yytext[yytextpos] = 0;
22 }
23
24
25 void yytextappend(int c)
26 {
27     if (yytextpos >= YYTEXT_MAX - 1)    //if textpos at the end of buffer
28     {
29         return; //ignore char
30     }
31     yytext[yytextpos++] = (char)c;
32     yytext[yytextpos] = 0;  //add null sentinel at end of buffer
33 }
34
35 int yylex(){
36     int c;
37
38     while (1){  //infinte while loop
39         yytextclear();  //clear out token buffer information
40
41         c = fin.get();  //retrives next character in file
42
43         //skip whitespaces since those are not part of tokens
44         while (c == ' ' || c == '\t' || c == '\n')
45         {
46             c = fin.get();  //c is now a char and not a whitespace (which isn't
                part of a token)
47         }
48
49         if (c == EOF )  //if end of file is reached
50         {
```

```
51              return EOFSY;
52          }
53
54          //Checking for comments /*........
55          if (followingCharsMatch(c, "/*"))   //if beginning of a section of
              comment
56          {
57              //while not end of comment "*/" or end of line (since comments must
                fit on single line)
58              while (!followingCharsMatch(c, "*/") && c != '\n')  //while end of
                  comment isn't reached or new line hasn't been reached, skip over
                  and ignore
59              {
60                  c = fin.get();  //retrieve next char
61              }
62              //if this point is reached, current character is either '\n' or end
                  of comment char "*"
63              continue;   //jumps to the outside while loop (while(1)) since
                  scanner should ignore comments
64          }
65
66          if (followingCharsMatch(c, "read"))
67          {
68              return READSY;
69          }
70
71          if (followingCharsMatch(c, "write"))
72          {
73              return WRITESY;
74          }
75
76          if (followingCharsMatch(c, ":="))
77          {
78              return ASSIGNOP;
79          }
80
81          yytextappend(c);
82
83          if (c == '+')
84          {
85              return ADDOP;
86          }
87
88          if (c == '-'){
89              return SUBOP;
90          }
91
92          if (c == '*') {
93              return MULTOP;
94          }
95
96          if (c == '/'){
```

```
 97              return DIVOP;
 98          }
 99
100          if (c == '('){
101              return LPAREN;
102          }
103
104          if (c == ')')
105          {
106              return RPAREN;
107          }
108
109
110          //Recognizing IDs
111          yytextclear();
112          if ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z'))
113          {
114              while ((c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z'))
115              {
116                  yytextappend(c);
117                  c = fin.get();
118              }
119              //once you reach this point, no longer an ID (letter)
120              fin.unget();     //returns the non ID char back
121              return ID;       //yytext should hold ID string whether it is "sum" or
                    "A"
122          }
123
124          //Recognizing numbers...if current char is a number
125          if (c >= '0' && c < '10')
126          {
127              while (c >= '0' && c < '10' || c == '.')   //while number or decimal
                    point (0-9)
128              {
129                  yytextappend(c);    //append that number to buffer
130                  c = fin.get();
131                  if (c == '.')   //if you come across a decimal point, keep
                        reading the rest of numbers
132                  {
133                      yytextappend(c);    //append to buffer
134                      c = fin.get();  //retrieve next char
135                      while (c >= '0' && c < '10')    //while it is a number, keep
                            appending to buffer
136                      {
137                          yytextappend(c);
138                          c = fin.get();
139                      }
140                  } //at this point, we have reached the end of numeric constants
141
142              }   //end while
143              fin.unget();    //unget char that wasn't number
144              return NUMCONST;
```

```
145         }
146         return c;   //unidentified char input...ignore
147     }
148     return EOFSY;   //should never get here
149 }
150
151 //checks to see if following characters match the inputted target...looks ahead
152 bool followingCharsMatch(int c, string target)
153 {
154     for (int i = 0; i < target.length(); i++)   //iterate through max length of
        target string's length
155     {
156         //tolower function returns lowercase equivalent to the parameter
            input...since scanner is case insensitive
157         if (tolower(c) == target.at(i)) //if current char is equal to first char
            of target string (case insensitive...so converted token to lower case)
158         {
159             yytextappend(c);
160             c = fin.get();
161         }
162         else //first chars don't match
163         {
164             yytextclear();  //clear token buffer
165             //unget characters that were retrieved to bring back to original
                state/location
166             while (i != 0)  //back to original location before comparing x
167             {
168                 fin.unget();    //put a character back
169                 i--;
170             }
171             return false;   //since we know characters/token doesn't match
172         }
173     }
174     fin.unget();    //unget first char that was retrieved to check for match
175     return true;    //if this point is reached, the current buffer and target is
        a match
176 }
177
178
```

```
/* terror.calc *    *  /  a1  /*  abc */
read A   /* Input the values ****/  xyz
read B23
/***/   12345)
/* bcd */ /* notatoken */
/*/  nope */

readabunch
writeread

/*****Perform the computations*****/
sum_total:=A+B
write sum
write sum_2be / 2.0 +  11.22.33.444
end
:
------------------------------------------------------------

angcharlie_35713_349284_CalcLex\CalcLex terror.calc
tok = 10 READSY (read)
tok = 08 ID (A)
tok = 08 ID (xyz)
tok = 10 READSY (read)
tok = 08 ID (B)
tok = 09 NUMCONST (23)
tok = 09 NUMCONST (12345)
tok = 02 RPAREN ())
tok = 10 READSY (read)
tok = 08 ID (abunch)
tok = 11 WRITESY (write)
tok = 10 READSY (read)
tok = 08 ID (sum)
tok = 09 NUMCONST (_total:=A)
tok = 03 ADDOP (+)
tok = 08 ID (B)
tok = 11 WRITESY (write)
tok = 08 ID (sum)
tok = 11 WRITESY (write)
tok = 08 ID (sum)
tok = 09 NUMCONST (_2be)
tok = 06 DIVOP (/)
tok = 09 NUMCONST (2.0)
tok = 03 ADDOP (+)
tok = 09 NUMCONST (11.22.33.444)
tok = 08 ID (end)
tok = 09 NUMCONST (:)
tok = 00 EOFSY--$$ ()

Number of tokens = 27
```

X B23

ID can have _

Num const only 1 . allowed

SUM_TOTAL

X

?