```
1
     #include<stdio.h>
 2
     #include<stdlib.h>
 3
     #include<string.h>
 4
     #define SIZE(x) (sizeof(x)/sizeof(x[0]))
 5
     //structure for each indivisual token
 6
     typedef struct stud1{
 7
     char name[100];
 8
     char type[100];
 9
     int occurences:
10
     int line no[10000];
11
     }token;
12
     //Symbol Table
13
     typedef struct stud2{
14
     token t[10000];
15
     int n:
16
     }symbol table;
17
     symbol table st;
18
     //to create a structure for each valid token in our language
19
     void verify(char *check,int k,char *w)
20
21
         strcpy(st.t[k].name,check);
22
         strcpy(st.t[k].type,w);
23
         st.t[k].occurences=0;
24
         st.t[k].line no[0]=0;
25
         st.n++;
26
     }
27
     //Declaring all various tokens in our defined language
28
     void valid tokens()
29
30
         char keyword[][20]={"auto", "extern", "sizeof", "break", "float", "static", "case",
                           "for", "struct", "char", "goto", "switch", "const", "if", "typedef",
31
                           "continue", "int", "union", "default", "long", "unsigned", "do",
32
                           "register", "void", "return", "double", "volatile", "else", "short",
33
                           "while", "enum", "signed"};
34
35
         char arithmetic[][20]={"+","-","*","/","%"};
         char relational[][20]={"<",">",">=","<=","==","!="};</pre>
36
37
         char conditional[][20]={"&&","||","!"};
38
         char assignment[][20]={"="};
         char inc dec[][20]={"++","--"};
39
         char punctuation[][20]={"{",",",";","}"," "};
40
41
         int k=0,i;
42
         //keyword token
         for(i=0;i<SIZE(keyword);i++)</pre>
43
44
45
           verify(keyword[i],k,"KEYWORD");
46
           k++;
47
         }
48
         //arithmetic token
49
         for(i=0;i<SIZE(arithmetic);i++)</pre>
50
         verify(arithmetic[i],k,"ARITHMETIC OPERATOR");
51
52
           k++;
53
         }
54
         //relational token
55
         for(i=0;i<SIZE(relational);i++)</pre>
56
57
         verify(relational[i],k,"RELATIONAL OPERATOR");
```

```
58
59
          }
60
           //conditional token
          for(i=0;i<SIZE(conditional);i++)</pre>
61
62
          verify(conditional[i],k,"CONDITIONAL OPERATOR");
63
64
             k++;
          }
65
66
          //assignment token
67
          for(i=0;i<SIZE(assignment);i++)</pre>
68
          verify(assignment[i],k,"ASSIGNMENT OPERATOR");
69
70
             k++;
71
          }
72
           //Increment-decrement token
          for(i=0;i<SIZE(inc dec);i++)</pre>
73
74
75
          verify(inc dec[i],k,"INC-DEC OPERATOR");
76
             k++;
77
          }
78
           //Puctuation token
79
          for(i=0;i<SIZE(punctuation);i++)</pre>
80
81
          verify(punctuation[i],k,"PUNCTUATION SYMBOL");
82
          k++;
83
84
      }
85
      void check(char* w,int line)
86
87
          int i,flag=0;
88
           //compare this word with all our existing tokens
89
          for(i=0;i<st.n;i++)</pre>
90
91
             if(!strcmp(w,st.t[i].name))
92
                break;
          }
93
94
           //Create a structure for this token if this word is not present in our defined \cline{2}
          set of tokens
95
          if(i!=st.n)
96
97
               st.t[i].line no[st.t[i].occurences]=line;
98
               st.t[i].occurences++;
99
          }
100
          else
101
           {
             if(!(w[0]>='0' \&\& w[0]<='9'))
102
103
                 //check for a Valid Identifier
104
105
               for(i=0;i<strlen(w);i++)</pre>
106
                 if((w[i]>='A' && w[i]<='Z')||(w[i]>='a' && w[i]<='z')||(w[i]=='_')||(w[i ⊋</pre>
107
                 ]>='0' \&\& w[i]<='9'))
108
                     flag=1;
109
                 else
110
                 {
111
                      flag=0;
112
                      break;
```

```
113
                 }
114
              }
115
               //Valid Token Identifier
              if(flag)
116
117
              {
118
                 strcpy(st.t[st.n].name,w);
119
                 strcpy(st.t[st.n].type,"IDENTIFIER");
120
                 st.t[st.n].line no[0]=line;
121
                 st.t[st.n].occurences=1;
122
                 st.n++;
123
              }
124
             }
125
            //Invalid Token Reporting ERROR
126
            if (!flag)
127
128
                 strcpy(st.t[st.n].name,w);
129
                 strcpy(st.t[st.n].type,"INVALID");
                 st.t[st.n].line_no[0]=line;
130
131
                 st.t[st.n].occurences=1;
132
                 st.n++;
133
            }
134
          }
135
136
      int other(char c)
137
138
        //Valid Symbols in our language other than alphanumeric characters
139
        char valid[]={'+','-','*','/','%','>','<','=','!','&','|'};</pre>
140
        int i:
141
        for(i=0;i<sizeof(valid);i++)</pre>
142
143
            if(valid[i]==c)
144
               return 1;
145
        }
146
        return 0;
147
      }
148
      int main()
149
      {
150
          int line=1,i,q=0,j,prev;
151
          char filename[100],s[100000],word[100000];
152
          st.n=0;
153
          printf("Enter the file name with.txt extension\n");
154
          scanf("%s", filename);
155
          FILE *fp=fopen(filename, "r");
156
          if(fp==NULL)
157
158
                 printf("Error File Cannot be opened\n");
159
                 exit(0);
160
161
          //Read file line by line
162
          valid tokens();
163
          while(fgets(s,sizeof(s),fp)!=NULL){
164
                   int flag=0;
165
                   q=0;
166
            for(i=0;i<strlen(s);i++)</pre>
167
168
                  //Check for an alphanumeric character
169
                   if(!flag &&((s[i]>='A' && s[i]<='Z')||(s[i]>='a' && s[i]<='z')||(s[i]>= ₹
```

```
'0' && s[i]<='9')||(s[i]==' ')))
170
171
                            word[q++]=s[i];
172
                         }
173
                   else
174
                        //For symbols like ++,--,<=,>= to be valid
175
176
                           if(flag && other(s[i]))
177
178
                                word[q++]=s[i];
179
180
                           else
181
                           {
182
                             //Not read tab character
                            if(s[i]!='\t')
183
184
                             {
185
                                  //Not read Null character
186
                                  if(q!=0)
187
                                   {
188
                                     word[q]='\setminus0';
189
                                     check(word,line);
190
191
                             }
                            q=0;
192
193
                            // All characters accepted other than tab ,carriage return and 📮
194
                            if(s[i]!='\t' && s[i]!='\n' && s[i]!='\r')
195
196
                              word[q++]=s[i];
197
                              //Special characters
198
                              if(!(other(s[i])))
199
                                  {
200
                                      //Present alphabet is an alphanumeric character
                                        if((s[i]>='A' && s[i]<='Z')||(s[i]>='a' && s[i]<='Z' ₹
201
                                        | | | (s[i] > = '0' \&\& s[i] < = '9') |
202
203
                                           flag=0;
204
                                        }
205
                                        else
206
                                      //Check if the word is a valid token
207
208
                                        word[q]='\setminus 0';
209
                                        check(word,line);
210
                                        q=0;
211
                                        flag=0;
212
                                        }
213
                                  }
214
                               else
215
                                  flag=1;
216
                              }
217
                            }
218
                        }
219
220
               //Increment line number
221
              line++;
222
223
            //Print Symbol Table
```

/home/himansu/Lexical Analyser.c
Page 5 of 5

Monday 25 August 2014 01:25:20 PM IST

```
224
           printf("\n\t\t\t\tSymbol Table\n");
225
           printf("\n\tTOKEN NAME \tTOKEN TYPE \t\tOCCURENCES\tLINE NUMBERS\n");
226
           for(i=0;i<(st.n);i++)</pre>
227
           {
228
                if(st.t[i].occurences>=1)
229
230
                    //if the token has occured then print it,its type, no of occurences
                                                                                                ₽
                    and line number
231
                    printf("\n^20s\t^20s\t^20s\t^210d",st.t[i].name,st.t[i].type,st.t[i].
                                                                                                ₽
                    occurences, st.t[i].line no[0]);
232
                    prev=st.t[i].line no[0];
233
                    for(j=1; j<st.t[i].occurences; j++)</pre>
234
                       {
235
                         if(!(prev==st.t[i].line no[j]))
236
                           printf(",%d",st.t[i].line no[j]);
237
238
                           prev=st.t[i].line no[j];
239
240
241
                    printf("\n");
242
                }
243
244
           return 0;
245
      }
246
247
```