

## lexicalanalyzer.c

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <ctype.h>
4  #include <string.h>
5
6  void main()
7  {
8      char ch, token[100][50];
9      int i = 0, j = 0, k;
10     FILE *f1, *f2;
11     f1 = fopen("input.txt", "r");
12     f2 = fopen("output.txt", "w");
13     if (f1 == NULL)
14     {
15         printf("Cannot open the input file.\n");
16         exit(0);
17     }
18     ch = getc(f1);
19     while (ch != EOF)
20     {
21         if (isalnum(ch))
22         {
23             if (isdigit(ch))
24             {
25                 while (isdigit(ch) && ch != EOF)
26                 {
27                     token[i][j++] = ch;
28                     ch = getc(f1);
29                 }
30             }
31             else
32             {
33                 while (isalnum(ch) && ch != EOF)
34                 {
35                     token[i][j++] = ch;
36                     ch = getc(f1);
37                 }
38             }
39             token[i][j] = '\0';
40             i++;
41             j = 0;
42         }
43         else if (!isspace(ch))
44         {
45             token[i][j++] = ch;
46             token[i][j] = '\0';
47             i++;
48             j = 0;
49             ch = getc(f1);
50         }
51         else if (isspace(ch))
```

```
52     {
53         while (isspace(ch) && ch != EOF)
54         {
55             ch = getc(f1);
56         }
57     }
58 }
59 for (k = 0; k < i; k++)
60 {
61     printf("%s", token[k]);
62     if (strcmp(token[k], "int") == 0 || strcmp(token[k], "main") == 0 ||
63     strcmp(token[k], "void") == 0)
64     {
65         printf("\tKeyword\n");
66         fprintf(f2, "%s\tKeyword\n", token[k]);
67     }
68     else if (strcmp(token[k], "+") == 0 || strcmp(token[k], "=") == 0)
69     {
70         printf("\tOperator\n");
71         fprintf(f2, "%s\tOperator\n", token[k]);
72     }
73     else if (strcmp(token[k], ";") == 0 || strcmp(token[k], "(") == 0 ||
74     strcmp(token[k], ")") == 0 ||
75     strcmp(token[k], "{") == 0 || strcmp(token[k], "}") == 0)
76     {
77         printf("\tSpecial character\n");
78         fprintf(f2, "%s\tSpecial character\n", token[k]);
79     }
80     else if (isdigit(token[k][0]))
81     {
82         printf("\tNumber\n");
83         fprintf(f2, "%s\tNumber\n", token[k]);
84     }
85     else if (isalpha(token[k][0]))
86     {
87         printf("\tIdentifier\n");
88         fprintf(f2, "%s\tIdentifier\n", token[k]);
89     }
90     else if (strcmp(token[k], "/") == 0)
91     {
92         if (strcmp(token[k + 1], "/") == 0)
93         {
94             printf("\n");
95         }
96         else
97         {
98             printf("\tOperator\n");
99             fprintf(f2, "%s\tOperator\n", token[k]);
100         }
101     }
102     else
103     {
104         printf("\tIdentifier\n");
105         fprintf(f2, "%s\tIdentifier\n", token[k]);
106     }
107 }
```

```
104         }
105     }
106     fclose(f1);
107     fclose(f2);
108 }
109
110
111
112
113 /*
114 ALGORITHM
115 1. Create a token array to store tokens and variables i, j and k for indexing
116 2. Open the input file in read mode and output file in write mode
117 3. If the input file cannot be opened, display an error and exit
118 4. Read the first character from input file
119 5. While there are characters left to read:
120     5.1 If the character is alphanumeric:
121         5.1.1 If it is a digit, collect full number
122         5.1.2 If it is a letter, collect full word or identifier
123     5.2 If the character is a non-space speacial symbol, store it as token
124     5.3 If the character is space, skip it
125 6. For each token:
126     6.1 Identify if it is a keyword
127     6.2 Identify if it is an operator
128     6.3 Identify if it is a speacial character
129     6.4 Identify numbers and identifiers
130 7. Print and save the tokens and tokens and their classifications to the output file
131 8. Close the input and output files
132 */
```