lexicalanalyzer.c

```c
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
#include <string.h>

void main()
{
    char ch, token[100][50];
    int i = 0, j = 0, k;
    FILE *f1, *f2;
    f1 = fopen("input.txt", "r");
    f2 = fopen("output.txt", "w");
    if (f1 == NULL)
    {
        printf("Cannot open the input file.\n");
        exit(0);
    }
    ch = getc(f1);
    while (ch != EOF)
    {
        if (isalnum(ch))
        {
            if (isdigit(ch))
            {
                while (isdigit(ch) && ch != EOF)
                {
                    token[i][j++] = ch;
                    ch = getc(f1);
                }
            }
            else
            {
                while (isalnum(ch) && ch != EOF)
                {
                    token[i][j++] = ch;
                    ch = getc(f1);
                }
            }
            token[i][j] = '\0';
            i++;
            j = 0;
        }
        else if (!isspace(ch))
        {
            token[i][j++] = ch;
            token[i][j] = '\0';
            i++;
            j = 0;
            ch = getc(f1);
        }
        else if (isspace(ch))
```

```
        {
            while (isspace(ch) && ch != EOF)
            {
                ch = getc(f1);
            }
        }
    }
    for (k = 0; k < i; k++)
    {
        printf("%s", token[k]);
        if (strcmp(token[k], "int") == 0 || strcmp(token[k], "main") == 0 ||
strcmp(token[k], "void") == 0)
        {
            printf("\tKeyword\n");
            fprintf(f2, "%s\tKeyword\n", token[k]);
        }
        else if (strcmp(token[k], "+") == 0 || strcmp(token[k], "=") == 0)
        {
            printf("\tOperator\n");
            fprintf(f2, "%s\tOperator\n", token[k]);
        }
        else if (strcmp(token[k], ";") == 0 || strcmp(token[k], "(") == 0 ||
strcmp(token[k], ")") == 0 ||
                 strcmp(token[k], "{") == 0 || strcmp(token[k], "}") == 0)
        {
            printf("\tSpecial character\n");
            fprintf(f2, "%s\tSpecial character\n", token[k]);
        }
        else if (isdigit(token[k][0]))
        {
            printf("\tNumber\n");
            fprintf(f2, "%s\tNumber\n", token[k]);
        }
        else if (isalpha(token[k][0]))
        {
            printf("\tIdentifier\n");
            fprintf(f2, "%s\tIdentifier\n", token[k]);
        }
        else if (strcmp(token[k], "/") == 0)
        {
            if (strcmp(token[k + 1], "/") == 0)
            {
                printf("\n");
            }
            else
            {
                printf("\tOperator\n");
                fprintf(f2, "%s\tOperator\n", token[k]);
            }
        }
        else
        {
            printf("\tIdentifier\n");
            fprintf(f2, "%s\tIdentifier\n", token[k]);
```

```
        }
    }
    fclose(f1);
    fclose(f2);
}



/*
ALGORITHM
1. Create a token array to store tokens and variables i, j and k for indexing
2. Open the input file in read mode and output file in write mode
3. If the input file cannot be opened, display an error and exit
4. Read the first character from input file
5. While there are characters left to read:
   5.1 If the character is alphanumeric:
       5.1.1 If it is a digit, collect full number
       5.1.2 If it is a letter, collect full word or identifier
   5.2 If the character is a non-space speacial symbol, store it as token
   5.3 If the character is space, skip it
6. For each token:
   6.1 Identify if it is a keyword
   6.2 Identify if it is an operator
   6.3 Identify if it is a speacial character
   6.4 Identify numbers and identifiers
7. Print and save the tokens and tokens and their classifications to the output file
8. Close the input and output files
*/
```