```
//************************************
//INCLUDE files for :Lexical Analyzer in C
//************************************
//stdio.h
//ctype.h
//string.h
//************************************
// Name: Lexical Analyzer in C
// Description:
// This program will lexically analyze a given file (C program), and identity the
//various tokens present in it.
// Input file: Need file name and its path.  Be sure the path is correct.
//
//************************************
#include<stdio.h>
#include<ctype.h>
#include<string.h>

void keyw(char *p);

int i = 0, id = 0, kw = 0, num = 0, op = 0, k=0;
char keywords[33][10] =
        {"auto", "main", "case", "char", "const",
         "continue", "default", "do", "double", "else",
         "enum", "extern", "float", "for", "goto",
         "if", "int", "long", "register", "return",
         "short", "signed", "sizeof", "static", "struct",
         "switch", "typedef", "union", "unsigned", "void",
         "volatile", "while"};

int main() {

    char ch;
    char str[25];
    char separators[15] = " \t\n,;(){}[]#\"<>";
    char operators[] = "!%^&*-+=~|.<>/?";
    char comments[2] = "//";
    int j;
    char fname[50] = "lab2_test.c";
    FILE *f1;

    f1 = fopen(fname, "r");

    if (f1 == NULL) {
        printf("file not found");
        //exit(0);
        return 0;
    }

    while ((ch = fgetc(f1)) != EOF) {

        if(ch=='/'){
            char buffer[100];
            fgets(buffer, 100, f1);
            continue;
        }

        //check for operators
        for (j = 0; j <= 14; j++) {
```

```
    if (ch == operators[j]) {
        {
            printf("%15c%15s\n", ch, "operator");
            op++;
            str[i] = '\0';
            //'0' - indicate the termination of a char string
            i = -1;
        }

    }
}
//check for separators
for (j = 0; j <= 14; j++) {
    if (i == -1)
        break;
    if (ch == separators[j]) {

        if (ch == '#') {
            while (ch != '>') {
                printf("%c", ch);
                ch = fgetc(f1);
            }
            printf("%15c%15s\n", ch, "header file");
            i = -1;
            break;
        }
        if (ch == '"') {
            do {
                ch = fgetc(f1);
                printf("%c", ch);
            } while (ch != '"');

            printf("\b argument\n");
            i = -1;
            break;
        }

        if(ch=='('){
            str[i] = '\0';
            keyw(str);
            printf("%15s%15s\n","(","lparen");
            break;
        }

        if(ch==')')
            printf("%15s%15s\n",")","rparen");

        if(ch=='{')
            printf("%15s%15s\n","{","lbrace");
        if(ch=='}')
            printf("%15s%15s\n","}","rbrace");
        if(ch==','){
            str[i] = '\0';
            keyw(str);
            printf("%15s%15s\n",",","comma");
            break;
        }
        if(ch==';'){
            str[i] = '\0';
```

```
                    keyw(str);
                    printf("%15s%15s\n",";","semicolon");
                    break;
                }



                str[i] = '\0';

                keyw(str);
            }

        }
        //Place the character to the string if it is not operator or separator
        if (i != -1) {
            str[i] = ch;
            i++;
        } else {
            i = 0;
        }
    }


}

void keyw(char *p) {

    int k, flag = 0;
    for (k = 0; k <= 31; k++) {
        if (strcmp(keywords[k], p) == 0) {

            printf("%15s%15s\n", p, "keyword");
            kw++;
            flag = 1;
            break;
        }
    }
    if (flag == 0) {
        if (isdigit(p[0])) {
            printf("%15s%15s\n", p, "number");
            num++;
        } else {
            //if(p[0]!=13&&p[0]!=10)
            if (p[0] != '\0' && p[0]!='\r') {

                printf("%15s%15s\n", p, "identifier");
                id++;
            }
        }

    }
    i = -1;
}
```