```c
/****************************************************************************************
*                              phraseFilterL
*****************************************************************************************

               phraseFilterL  <input.txt>  <output.txt>  <lim>"

Reads a text file, puts all phrases (between 2 consecutive full stops) in the rows of a
2D array. If a phrase contains more words than given by 'lim', the phrase is copied in a
text file. If no phrase has more than 'lim' words a message is printed.
*/


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

# define MAX 1000

int FtoM (FILE *, char [][MAX]);
int words (char []);


int main (int argc, char * argv[]) {
    if (argc != 4) {
        printf("\nInvalid Command!\n");
        return 0;
    }

    if (*(argv + 3)[0] == '-') {
        printf("\nError: negative number not allowed!\n");
        return 0;
    }

    unsigned short lim;
    lim = atoi (*(argv + 3));

    FILE *fin, *fout;
    char m[MAX][MAX];
    int rows, numWords, numPhrases=0;

    fin  = fopen (*(argv + 1), "r");
    fout = fopen (*(argv + 2), "w");

    rows = FtoM (fin, m);

    fprintf(fout, "Phrases with more than %d words:\n\n", lim);

    for (int i = 0; i < rows; i++) {
        if ((numWords = words(m[i])) > lim) {
            fprintf(fout, "Phrase n. %d:\n%s (%d words) \n\n", (i + 1), m[i], numWords);
            numPhrases++;
        }
    }

    if (numPhrases == 0)
        fprintf (fout, "None.\n\n");

    printf ("\nRead results in %s\n\n", *(argv + 2));

    fclose(fin);
    fclose(fout);

    return 0;
}




/****************************************************************************************
*                                     words
*****************************************************************************************

Counts the number of words in a string containing a phrase.

It performs the task by counting the number of empty spaces or '\', '/', and apostrophes
between words; this counting incremented by 1 is the number of words.
In case of consecutive empty spaces (' ', '\t'), only 1 is counted.

The enumation 'st' is used to detect whether we're inside a word (in) or not (out).
The count of word separators is incremented when one is read when st=in. Word separators
read when st=out are ignored.
If st=out and alphannumeric characters are read, 'st' is set to 'in'.
If st=in and alphannumeric characters are read, they are ignored.

NB: If there are multiple empty spaces before the end of the string, 'st' will be equal to
    'out' and 1 more space would be counted. So there's some correction at the end.

NB: It's assumed that there is no empty space at the beginning of the string.

NB: '-' and '_' are assumed to be used only to merge 2 words together into 1 word.
*/



int words (char s[]) {
    unsigned short separators = 0;
    enum st {out, in} st;

    st = in;

    for (int i = 0; s[i] != '\0'; i++) {
        if ((isspace(s[i]) || s[i] == '\'' || s[i] == '\\' || s[i] == '/')
            && st == in) {
            separators++;
            st = out;
        }
        else if ((isspace(s[i]) || s[i] == '\'' || s[i] == '\\' || s[i] == '/')
                 && st == out)
            continue;
        else if (isalnum(s[i]) && st == out)
            st = in;
        /* Alphannumerical characters or punctuation AND st=in*/
        else
            continue;
    }

    if (st == in)
        return separators + 1;
    else
        return separator;

}





/*****************************************************************************************
*                                         FtoM
******************************************************************************************

Receives a pointer to a file and a 2D array of char.
Copies each sentence in a row of the array and returns the number of rows that have been
filled, which is the number of sentences.
It ignores empty spaces at the beginning.

To determine the end of a sentence it checks that an empty space, a new line or a
tabulation character follows a full stop character (the latter is copied too). A new line
character in the text is translated into an empty space in the array. If other characters
follow a full stop character, it continues to copy in the same sentence.

BEWARE: other types of mistakes (e.g. punctuation characters after '.') are not checked.
*/

int FtoM (FILE *f, char m[][max]) {
    int c, i=0, j=0;

    while ((c = getc(f)) != EOF && j < max) {
        if (isspace(c) && j == 0)
            continue;
        else if (c != '.' && c != '\n')
            m[i][j++] = c;
        else if (c == '.') {
            m[i][j++] = c;

            if ( isspace( (c = getc(f)) ) ) {
                m[i++][j] = '\0';
                j = 0;
            }
            else if (c == EOF) {
                m[i][j] = '\0';
                break;
            }
            else
                m[i][j++] = c;
        }
        else
            m[i][j++] = ' ';
    }
    return i + 1;
}
```