```c
/*********************************************************************
*                              countWordsN
*********************************************************************/
/*
             contaParole  <input.txt>  <output.txt>  <n>

Reads a text file and prints on another text file the number of words of at least
n characters (n must be greater or equal to 1).
*/


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


unsigned readerOne (FILE *);
int reader (FILE *, int);


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

    if (*(argv + 3)[0] == '-') {
        printf("\nERROR.\n");
        return 0;
    }

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

    FILE *fin;
    FILE *fout;

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

    /* Two functions are used for the cases n=1 and n>1 because for n=1 the
       algorithm is simpler */
    if (n == 1)
        fprintf(fout, "N. of words at least 1 character long: %d", readerOne(fin));
    else
        fprintf(fout, "N. of words at least %d character long: %d", n, reader(fin, n));

    fclose (fin);
    fclose (fout);

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

}
```

```c
/*****************************************************************************
*                                readerOne
*****************************************************************************

Counts the number of words with at least 1 character. The function has direct
access to the input text file.

I use 'st' to detect whether I'm reading a word (in) or not (out).

Empty spaces and punctuation marks are ignored when st=out.
When st=out and alphanumerical characters are read, 'st' is set to 'in'.
'st' remains in the same state as long as alphanumerical characters come.
When st=in and an empty space or an apostrophe are read, a word is counted and
'st' is set to 'out'.

Some care is needed for the last word: if the last word is immidiately followed
by EOF 'st' remains 'in' and the last word wouldn't be counted. So a check at
the end makes a correction in that case.

NB: Special characters are not considered.
NB: Wrong characters at the beginning or at the end of a word are not treated.
*/

unsigned readerOne (FILE *f) {
    unsigned words = 0, c;
    enum st{out, in} st;

    st = out;

    while ((c = getc(f)) != EOF) {
        if ( st == out && (isspace(c) || ispunct(c)) )
            continue;
        else if ( st == in && ( isspace(c) || ispunct(c)) ) {
            st = out;
            words++;
        }
        else if ( st == out && !( isspace(c) || ispunct(c)) )
            st = in;
        /* Nothing to do in other cases */
        else
            continue;
    }

    if (st == out)
        return words;
    else
        return (words + 1);
}
```

```c

/*****************************************************************************
*                                Reader
*****************************************************************************/
/*
Counts words with at least n characters, reading from the input text file

Punctuation characters are ignored.
If it reads alphannumerical characters, it increments 'letters'.
Empty spaces at the beginning or end of file (letters=0) are ignored.
When empty spaces or puntuation marks are read immediately after the end of a
word (letters > n-1), 'words' is incremented and 'i' is reset to 0. If the same
happens when 0 < letters < n-1, the word read is too short and is not counted
but 'letters' is reset to 0.
As for the other function, the last word requires some care.
*/


int reader (FILE * f, int n) {
    int c;
    unsigned short letters = 0;
    unsigned short words   = 0;
        while ((c = getc(f)) != EOF) {
            if ((isspace(c) || ispunct(c)) && letters == 0)
                continue;
            else if ((isspace(c) || ispunct(c)) && letters > (n - 1)) {
                letters = 0;
                words++;
            }
            else if ((isspace(c) || ispunct(c)) && letters > 0 && letters <= (n - 1))
                letters = 0;
            else if ( (c != '\'') && ispunct(c) )
                continue;
            else
                letters++;
        }

    if (letters > n - 1)
        words++;

    return words;
}
```