

```
1  /*****
2  *                               countPalindrome
3  *****/
4  Command format:
5
6      contaPalindromi <input.txt> <output.txt> <row>
7
8  Receives sentences from <input.txt> and writes the number of palindrome words in
9  the sentence indicated by <row>, and prints the sentence itself.
10
11  It checks the correctness of the command format (number of arguments passed and
12  value of <row>).
13  */
14
15  # include <stdio.h>
16  # include <string.h>
17  # include <stdlib.h>
18  # include <ctype.h>
19
20  # define MAX 1000
21
22  typedef unsigned short int U_S_INT;
23
24  unsigned int palCount(char []);
25  unsigned int FtoM(FILE *, char[][MAX]);
26  U_S_INT isPalindrome_1(char []);
27  U_S_INT isPalindrome(char[], int, int);
28
29
30
31  int main (int argc, char * argv[]) {
32      if (argc != 4){
33          printf("\n Incorrect command!\n");
34          return 0;
35      }
36
37      int rows;
38      char m[MAX][MAX];
39      U_S_INT sentence;
40      FILE *fin, *fout;
41
42      fin = fopen(*(argv + 1), "r");
43      fout = fopen(*(argv + 2), "w");
44
45      sentence = atoi(*(argv + 3));
46      rows = FtoM(fin , m);
47
48      if (sentence == 0) {
49          printf("\nInvalid Argument (n. of Sentence)\n");
50          return 0;
51      }
52      else if (sentence > rows) {
53          printf("\nNot enough sentences in %s\n", *(argv + 1));
54          return 0;
55      }
56
57      sentence--;
58
```

```

59
60     fprintf (fout, "%s\n\nSentence n.: %d\n\n", m[sentence], (sentence + 1));
61     fprintf (fout, "N. of Palindrome words: %d\n\n", palCount(m[sentence]) );
62     fprintf (fout, "END.\n");
63
64     printf("\nResults in %s\n", *(argv + 2));
65
66     fclose(fin);
67     fclose(fout);
68 }
69
70
71 /*****
72  *
73  *****/
74
75 Reads 'r' and writes words into 'w'; then checks that 'w' is palindrome
76 with the function 'isPalindrome' and if it's the case, 'palindromes' is
77 incremented. The variable 'j' is used to detect when we're into a word.
78
79 NB: the function assumes that characters '-' and '_' can only be part of a word
80 and that '.' can be part of a word or at the end of a sentence.
81
82 NB: The presence of '\n' is not expected.
83 */
84
85
86 unsigned int palCount (char r[]) {
87     U_S_INT j = 0;
88     int i = 0;
89     int palindromes = 0;
90     char w[MAX];
91
92     while(r[i] != '\0') {
93         if (j == 0 && (ispunct(r[i]) || isspace(r[i])))
94             continue;
95         else if ( j != 0
96                 && (isspace(r[i]) || (ispunct(r[i])
97                                     && (r[i] != '-') && (r[i] != '_')
98                                     && (r[i] != '.'))))) {
99             w[j] = '\0';
100             if (isPalindrome(w, 0, j - 1))
101                 palindromes++;
102             j = 0;
103         }
104         else if (j != 0 && (r[i] == '.' && r[i + 1] == '\0')) {
105             w[j] = '\0';
106             if (isPalindrome(w, 0, j - 1))
107                 palindromes++;
108             j = 0;
109         }
110         else
111             w[j++] = r[i];
112         i++;
113     }
114     return palindromes;
115 }
116

```

```
117
118 /*****
119 *                               isPalindrome
120 *****/
121
122 /* Two versions, one recursive, one iterative.*/
123
124 U_S_INT isPalindrome(char s[], int i, int j) {
125     if (s[i] == s[j] && i < j)
126         isPalindrome(s, i + 1, j - 1);
127     else
128         return 0;
129     return 1;
130 }
131
132
133 U_S_INT isPalindrome_1(char s[]) {
134     int i = 0, len, range;
135
136     if ((len = strlen(s)) <= 2)
137         return 0;
138     else if (len % 2 != 0)
139         range = (len - 1) / 2;
140     else
141         range = len / 2;
142
143     len--;
144
145     while (s[i] == s[len - i] && i < range)
146         i++;
147
148     if (i == range)
149         return 1;
150     else
151         return 0;
152 }
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
```

```
175 /*****
176 *                               FtoM
177 *****/
178
179 unsigned int FtoM (FILE *f, char m[][MAX]) {
180     int c, i=0, j=0;
181
182     while ((c = getc(f)) != EOF && j < MAX) {
183         /* Ignore initial spaces */
184         if (isspace(c) && j == 0)
185             continue;
186         else if (c != '.' && c != '\n')
187             m[i][j++] = c;
188         else if (c == '.') {
189             m[i][j++] = c;
190
191             if ( isspace( (c = getc(f)) ) ) {
192                 m[i++][j] = '\0';
193                 j = 0;
194             }
195             else if (c == EOF) {
196                 m[i][j] = '\0';
197                 break;
198             }
199             else
200                 m[i][j++] = c;
201         }
202         else
203             m[i][j++] = ' ';
204     }
205     return i + 1;
206 }
207
```