ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ

**Федеральное государственное автономное образовательное учреждение
высшего образования
«Национальный исследовательский университет
«Высшая школа экономики»**

Московский институт электроники и математики им. А.Н. Тихонова

Департамент прикладной математики

Отчёт
по лабораторной работе №7
по курсу «Алгоритмизация и программирование»
Задание № 13

ФИО студента	Номер группы	Дата
Кейер Александр Петрович	БПМ-231	13 января 2024 г.

Москва, 2023

Задание (вариант № 13)

Написать функцию обработки строки и программу, тестирующую данную функцию. В программе должен быть предусмотрен вывод исходный строки, которая при выделении слов не должна измениться.

Дана строка, содержащая от 1 до 30 слов, в каждом из которых от 1 до 10 латинских букв и/или цифр; между соседними словами– запятая, за последним словом – точка. Напечатать эту же последовательность слов, но удалив из нее повторные вхождения слов.

Решение 1 (аккуратно используем strtok)

```c
#include <stdio.h> // Input/outpu library.
#include <string.h> // String libarary for special function : strtok, strcat etc.

// Useful directives for words.
#define maxWordsCount 30
#define minWordsCount 1
#define maxWordLength 10
#define minWordLength 1

// Useful directives for string.
#define maxSLength maxWordsCount * maxWordLength
#define minSLength minWordsCount * minWordLength

// Useful directives for string tokens.
#define separator ","
#define endToken "."

// Function validating a string.
int isSValid(char s[maxSLength]) {
  size_t sLength = strlen(s);

  char* endTokenP = strstr(s, endToken);

  // Checking the last symbol for a dot.
  if (endTokenP != s + sLength - 1) {
    printf("The string must have no more than 300 symbols, contain only one dot and end with this dot.\n");
    return 0;
  }

  // Checking string length.
  if (sLength > maxSLength || sLength < minSLength) {
    printf("Incorrect string length.\n");
    return 0;
  }

  // Checking string for incorrect symbols.
  for (int i = 0; s[i] != '\0'; i++) {
    if (!(
    s[i] == 44
    || s[i] == 10
    || s[i] == 46
```

```
42        || (s[i] >= 65 && s[i] <= 90)
43        || (s[i] >= 97 && s[i] <= 122)
44        || (s[i] >= 48 && s[i] <= 57)
45        )) {
46          printf("The string contains incorrect symbols.\n");
47          return 0;
48        }
49      }
50
51      return 1;
52    }
53
54    // Function reading a string from stdin.
55    int sReading(char sExternal[maxSLength]) {
56      printf("Please, enter correct string:\n");
57
58      char s[maxSLength];
59      fseek(stdin, 0, SEEK_END); // Jumped over previous stdin.
        We also can clear stdin here: fflush(stdin);
60      fgets(s, maxSLength, stdin);
61
62      s[strlen(s) - 1] = '\0'; // Remove \n symbol.
63      strcpy(sExternal, s);
64
65      return 0;
66    }
67
68    // Function printing words array.
69    int printWordsArr(char wordsArr[maxWordsCount][
       maxWordLength], int wordsCount) {
70      for (int i = 0; i < wordsCount; i++) {
71        for (int j = 0; j < maxWordLength; j++) {
72          if (wordsArr[i][j] == '\0') {
73            break;
74          }
75
76          printf("%c", wordsArr[i][j]);
77        }
78
79        // Placing tokens correctly.
80        if (i < wordsCount - 1) {
81          printf(",");
82        } else {
83          printf(".\n");
84        }
```

```
85      }
86
87      return 0;
88    }
89
90    // Function changing matrix row.
91    int changeMatrixRow(char matrix[maxWordsCount][
       maxWordLength], int i, char row[maxWordLength]) {
92      for (int j = 0; j < maxWordLength; j++) {
93        matrix[i][j] = row[j];
94      }
95
96      return 0;
97    }
98
99    // Function checking for the presence of a row in array.
100   int isMatrixContainRow(char matrix[maxWordsCount][
       maxWordLength], int rowsCount, char row[maxWordLength]) {
101     for (int i = 0; i < rowsCount; i++) {
102       for (int j = 0; j < maxWordLength; j++) {
103         if (matrix[i][j] != row[j]) {
104           break;
105         }
106
107         // Checking for the end token.
108         if (matrix[i][j] == '\0' || j == maxWordLength - 1) {
109           return 1;
110         }
111       }
112     }
113
114     return 0;
115   }
116
117   // Function presenting solution.
118   void solution(char s[maxSLength]) {
119     printf("You entered string: %s\n", s);
120
121     // String validation.
122     if (!isSValid(s)) {
123       return;
124     }
125
126     s[strlen(s) - 1] = '\0'; // Remove . symbol.
127
```

```
128     // Useful variables initialization.
129     char* word;
130     int uniqueWordsCount = 0;
131     int wordsCount = 0;
132     char wordsArr[maxWordsCount][maxWordLength];
133 
134     word = strtok(s, separator);
135 
136     // Checking for correctly first word.
137     if (word == NULL) {
138       printf("Incorrect word length or words count.\n");
139       return;
140     }
141 
142     // Splitting string into words.
143     while (word != NULL) {
144       size_t wordLength = strlen(word);
145 
146       // Checking for correctly rest words.
147       if (
148       wordLength < minWordLength
149       || wordLength > maxWordLength
150       || wordsCount > maxWordsCount
151       ) {
152         printf("Incorrect word length or words count.\n");
153         return;
154       }
155 
156       // Adding word into special array.
157       if (!isMatrixContainRow(wordsArr, uniqueWordsCount,
    word)) {
158         changeMatrixRow(wordsArr, uniqueWordsCount++, word);
159       }
160 
161       wordsCount++;
162       word = strtok(NULL, separator);
163     }
164 
165     printf("New string: ");
166 
167     // Printing special word array.
168     printWordsArr(wordsArr, uniqueWordsCount);
169   }
170 
171   // Main function.
```

```
172 int main() {
173
174   // Test 1.
175   printf("Test 1\n");
176   char s1[] = "a,a.";
177   solution(s1);
178   printf("\n===\n\n");
179
180   // Test 2.
181   printf("Test 2\n");
182   char s2[] = "a,a";
183   solution(s2);
184   printf("\n===\n\n");
185
186   // Test 3.
187   printf("Test 3\n");
188   char s3[] = "a.a.";
189   solution(s3);
190   printf("\n===\n\n");
191
192   // Test 4.
193   printf("Test 4\n");
194   char s4[] = ".";
195   solution(s4);
196   printf("\n===\n\n");
197
198   // Test 5.
199   printf("Test 5\n");
200   char s5[] = "asdasdasdas.";
201   solution(s5);
202   printf("\n===\n\n");
203
204   // Test 6.
205   printf("Test 6\n");
206   char s6[] = "a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,
    a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,
    a,a,a,a,a,a,a,a,a,a.";
207   solution(s6);
208   printf("\n===\n\n");
209
210   // Test 7.
211   printf("Test 7\n");
212   char s7[] = "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa,
    aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa,
    aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa,
```

```
    aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa,
    aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa,
    aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa,
    aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa.";
    solution(s7);
    printf("\n===\n\n");

    // User test.
    char s8[] = "";
    sReading(s8);
    solution(s8);

    return 0;
}

```

Решение 2 (аккуратно не используем strtok)

```c
#include <stdio.h> // Input/outpu library.
#include <string.h> // String libarary for special function : strtok, strcat etc.

// Useful directives for words.
#define maxWordsCount 30
#define minWordsCount 1
#define maxWordLength 10
#define minWordLength 1

// Useful directives for string.
#define maxSLength maxWordsCount * maxWordLength
#define minSLength minWordsCount * minWordLength

// Useful directives for string tokens.
#define separator ","
#define endToken "."

// Function validating a string.
int isSValid(char s[maxSLength]) {
  size_t sLength = strlen(s);

  char* endTokenP = strstr(s, endToken);

  // Checking the last symbol for a dot.
  if (endTokenP != s + sLength - 1) {
    printf("The string must have no more than 300 symbols, contain only one dot and end with this dot.\n");
    return 0;
  }

  // Checking string length.
  if (sLength > maxSLength || sLength < minSLength) {
    printf("Incorrect string length.\n");
    return 0;
  }

  // Checking string for incorrect symbols.
  for (int i = 0; s[i] != '\0'; i++) {
    if (!(
    s[i] == 44
    || s[i] == 10
    || s[i] == 46
```

```
      || (s[i] >= 65 && s[i] <= 90)
      || (s[i] >= 97 && s[i] <= 122)
      || (s[i] >= 48 && s[i] <= 57)
      )) {
        printf("The string contains incorrect symbols.\n");
        return 0;
      }
    }

    return 1;
  }

  // Function reading a string from stdin.
  int sReading(char sExternal[maxSLength]) {
    printf("Please, enter correct string:\n");

    char s[maxSLength];
    fseek(stdin, 0, SEEK_END); // Jumped over previous stdin.
    We also can clear stdin here: fflush(stdin);
    fgets(s, maxSLength, stdin);

    s[strlen(s) - 1] = '\0'; // Remove \n symbol.
    strcpy(sExternal, s);

    return 0;
  }

  // Function printing words array.
  int printWordsArr(char wordsArr[maxWordsCount][
   maxWordLength], int wordsCount) {
    for (int i = 0; i < wordsCount; i++) {
      for (int j = 0; j < maxWordLength; j++) {
        if (wordsArr[i][j] == '\0') {
          break;
        }

        printf("%c", wordsArr[i][j]);
      }

      // Placing tokens correctly.
      if (i < wordsCount - 1) {
        printf(",");
      } else {
        printf(".\n");
      }
```

```
85      }
86
87      return 0;
88    }
89
90    // Function changing matrix row.
91    int changeMatrixRow(char matrix[maxWordsCount][
       maxWordLength], int i, char row[maxWordLength]) {
92      for (int j = 0; j < maxWordLength; j++) {
93        matrix[i][j] = row[j];
94      }
95
96      return 0;
97    }
98
99    // Function checking for the presence of a row in array.
100   int isMatrixContainRow(char matrix[maxWordsCount][
       maxWordLength], int rowsCount, char row[maxWordLength]) {
101     for (int i = 0; i < rowsCount; i++) {
102       for (int j = 0; j < maxWordLength; j++) {
103         if (matrix[i][j] != row[j]) {
104           break;
105         }
106
107         // Checking for the end token.
108         if (matrix[i][j] == '\0' || j == maxWordLength - 1) {
109           return 1;
110         }
111       }
112     }
113
114     return 0;
115   }
116
117   // Function presenting solution.
118   void solution(char s[maxSLength]) {
119     printf("You entered string: %s\n", s);
120
121     // String validation.
122     if (!isSValid(s)) {
123       return;
124     }
125
126     s[strlen(s) - 1] = '\0'; // Remove . symbol.
127
```

```
128     // Useful variables initialization.
129     char word[maxWordLength] = "";
130     int uniqueWordsCount = 0;
131     int wordsCount = 0;
132     char wordsArr[maxWordsCount][maxWordLength];
133     char symbol[2] = {'\0', '\0'};
134 
135     // Checking for correctly first word.
136     if (word == NULL) {
137       printf("Incorrect word length or words count.\n");
138       return;
139     }
140 
141     // Splitting string into words.
142     for (int i = 0; i < maxSLength; i++) {
143       symbol[0] = s[i];
144 
145       if (symbol[0] == '\0') {
146         break;
147       }
148 
149       // Adding a symbol.
150       if (symbol[0] != ',') {
151         strcat(word, symbol);
152       }
153 
154       size_t wordLength = strlen(word);
155 
156       // Checking for correctly rest words.
157       if (
158       wordLength < minWordLength
159       || wordLength > maxWordLength
160       || wordsCount > maxWordsCount
161       ) {
162         printf("Incorrect word length or words count.\n");
163         return;
164       }
165 
166       // Adding word into special array.
167       if (symbol[0] == ',') {
168         if (!isMatrixContainRow(wordsArr, uniqueWordsCount,
    word)) {
169           changeMatrixRow(wordsArr, uniqueWordsCount++, word)
    ;
170         }
```

```
171
172         // Word reset.
173         word[0] = '\0';
174       }
175
176       wordsCount++;
177     }
178
179     // Printing special word array.
180     printf("New string: ");
181     printWordsArr(wordsArr, uniqueWordsCount);
182   }
183
184   // Main function.
185   int main() {
186
187     // Test 1.
188     printf("Test 1\n");
189     char s1[] = "a,a.";
190     solution(s1);
191     printf("\n===\n\n");
192
193     // Test 2.
194     printf("Test 2\n");
195     char s2[] = "a,a";
196     solution(s2);
197     printf("\n===\n\n");
198
199     // Test 3.
200     printf("Test 3\n");
201     char s3[] = "a.a.";
202     solution(s3);
203     printf("\n===\n\n");
204
205     // Test 4.
206     printf("Test 4\n");
207     char s4[] = ".";
208     solution(s4);
209     printf("\n===\n\n");
210
211     // Test 5.
212     printf("Test 5\n");
213     char s5[] = "asdasdasdas.";
214     solution(s5);
215     printf("\n===\n\n");
```

```
216
217     // Test 6.
218     printf("Test 6\n");
219     char s6[] = "a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,
    a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,
    a,a,a,a,a,a,a,a,a,a,a.";
220     solution(s6);
221     printf("\n===\n\n");
222
223     // Test 7.
224     printf("Test 7\n");
225     char s7[] = "aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa,
    aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa,
    aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa,
    aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa,
    aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa,
    aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa,
    aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa.";
226     solution(s7);
227     printf("\n===\n\n");
228
229     // User test.
230     char s8[] = "";
231     sReading(s8);
232     solution(s8);
233
234     return 0;
235 }
236
```

Тесты

```
You entered string: a,a.
New string: a.

===

Test 2
You entered string: a,a
The string must have no more than 300 symbols, contain only one dot and end with this dot.

===

Test 3
You entered string: a.a.
The string must have no more than 300 symbols, contain only one dot and end with this dot.

===

Test 4
You entered string: .
Incorrect word length or words count.

===

Test 5
You entered string: asdasdasdas.
Incorrect word length or words count.

===

Test 6
You entered string: a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a,a.
Incorrect word length or words count.

===

Test 7
You entered string: aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa.
Incorrect string length.
```