**Правительство Российской Федерации**

**Федеральное государственное автономное образовательное учреждение высшего профессионального образования   
"Национальный исследовательский университет   
"Высшая школа экономики"**

Московский институт электроники и математики Национального

исследовательского университета "Высшая школа экономики"

Департамент прикладной математики

**ОТЧЕТ**

**По лабораторной работе №7**

**ОБРАБОТКА СТРОК**

**По курсу «Алгоритмизация и программирование»**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| |  |  | | --- | --- | |  | ФИО студента | | Номер группы | Дата |
| Кармаев Александр Андреевич | БПМ211 | 15.10.2021 |
|  |
|  |
|  |

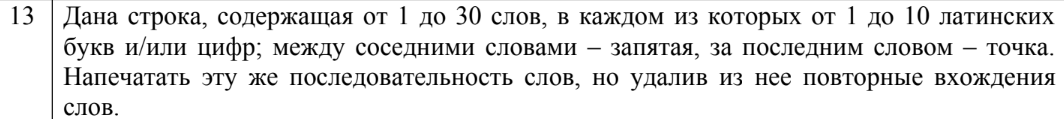
**Москва – 2021 г.**

**ЗАДАНИЕ (вариант №13)**

*Написать функцию обработки строки и программу, тестирующую данную*

*функцию. В программе должен быть предусмотрен вывод исходной строки,*

*которая при выделении слов не должна измениться.*

****

**РЕШЕНИЕ**

#include <stdio.h>

#include <stdlib.h>

#include <string.h> // библиотека для работы со строками(массивами символов)

int contain(char ch) { // функция проверяет, является ли символ допустимым для использования (латинские буквы и цифры)

char aval[] = "qwertyuiopasdfghjklzxcvbnmQWERTYUIOPASDFGHJKLZXCVBNM1234567890"; // массив допустимых символов

int aval\_length = strlen(aval); // подсчитываем длинну строки aval c помощью функции из библиотеки string.h

for (int i = 0; i < aval\_length; i++) {

if (aval[i] == ch) // если находим совпадение, то возвращаем 1 (правда - символ является допустимым)

return 1;

}

return 0; // если не будет возвращена 1, то возвращаем 0 (ложь)

}

int condition\_and\_answer(const char\* str) { // функция возвращает "ложь", если строка не соответствует условию задачи // а если соответствует, то будет напечатана итоговая строка в соответствии с условием задачи

int str\_length = strlen(str);

if (str[str\_length - 1] != '.') // если в конце нет точки, то дальше можно и не смотреть

return 1;

char\* word = NULL; // динамическая строка, в которую будем записывать наши слова

int word\_len = 0; // длинна слов

char\*\* arr = NULL; // динамический массив строк, в который будем добавлять наши слова, чтобы потом было удобно напечатать финальную строку, в которой не будет повторяющихся слов

int arr\_len = 0; // длинна массива слов

for (int i = 0; i < str\_length; i++) { // пробегаемся по каждому символу изначальной строки

if (contain(str[i])) { // если символ допустим, то мы добавляем его в наше слово

word\_len++;

if (word\_len > 10) { // проверка на то, что слова состоят меньше, чем из 10 символов

return 1;

}

word = realloc(word, (word\_len + 1) \* sizeof(char));

word[word\_len - 1] = str[i];

word[word\_len] = '\0'; // присваиваем последнему элементу слова символ конца строки для дальнейшей корректной работы при печати слов и их сравнении

}

else if (str[i] == ',' || str[i] == '.') { // если символ недопустим, но это точка или запятая, то мы добавляем слово в массив слов

if (str[i] == '.' && i != str\_length - 1) // проверяем наличие нескольких точек в строке

return 1;

if (word\_len == 0) // проверяем, что слово состоит хотя бы из одного символа

return 1;

arr\_len++;

if (arr\_len > 30) // проверка на то, что строка состоят меньше, чем из 30 слов

return 1;

arr = realloc(arr, arr\_len \* sizeof(char\*));

arr[arr\_len - 1] = malloc((word\_len + 1) \* sizeof(char));

arr[arr\_len - 1] = word;

word = NULL; // обнуляем наше слово, чтобы записывать в него новые

word\_len = 0;

}

else // встреча недопустимого символа

return 1;

}

// если строка введена корректно, то выводим её в соответствии с условием

printf("Final string: ");

printf("%s", arr[0]); // выводим первое слово, чтобы упростить печать запятых и точки

for (int i = 1; i < arr\_len; i++) { // пробегаемся по каждому слову из массива слов

int f = 1; // флаг, отвечающий за нужду печатать слово

for (int j = 0; j < i; j++) { // пробегаемся по словам предшествующим текущему слову с индексом = i

if (!strcmp(arr[i], arr[j])) { // если слова одинаковы, то завершаем цикл проверки и присваиваем флагу 0, чтобы слово не напечаталось

f = 0;

break;

}

}

if (f) // если слово ещё не было напечатано, то делаем это

printf(",%s", arr[i]);

}

printf(".\n");

return 0;

}

int main(void) {

char str[331]; // создаём строку с максимальным допустимым размером 30 слов \* 10 символов в слове + 30 возможных знаков препинания + символ конца строки

do{

printf("Input string by task condition: ");

gets(str); // считываем данные в строку c помощью функции из stdio.h

if (strlen(str) == 331 && str[strlen(str) - 1] != "\0") {

printf("You break tne main condition in inputing string!!\nGoing beyond the boundaries of the array!!");

exit(EXIT\_FAILURE); // вызываем exit() вместо return, так как при выходе за границу массива стек около него будет повреждён, что не позволит программе дальше корректно работать

}

}

} while (condition\_and\_answer(str)); // просим пользователя ввести строку, пока она не будет введена в соответствии с условием задачи

printf("The original string: %s \n", str); // выводим изначальную строку

return 0;

}

**ТЕСТЫ**

**Тест № 1** // пример содержит обработку ошибки ввода недопустимого символа

Изображение выглядит как текст

Автоматически созданное описание

**Тест № 2**

Изображение выглядит как текст

Автоматически созданное описание

**Тест № 3**

Изображение выглядит как текст

Автоматически созданное описание

**Тест № 4 // одно слово – нет запятых**

Изображение выглядит как текст

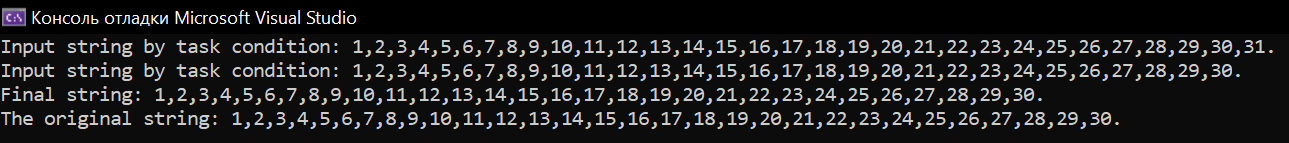
Автоматически созданное описание

**Тест № 5 // слово состоит более чем из 10 символов**

Изображение выглядит как текст

Автоматически созданное описание

**Тест № 6 // строка состоит более чем из 30 слов**



**Тест № 7 // строка состоит более чем из 330 символов(выход за границы массива)**

Изображение выглядит как текст

Автоматически созданное описание

**Тест № 8 //** проверяем наличие нескольких точек в строке (ошибка без доп. проверки могла бы возникнуть из-за специфики выбранного мною алгоритма действий)Изображение выглядит как текст

Автоматически созданное описание