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

Московский институт электроники и математики им. А.Н. Тихонова НИУ ВШЭ
Департамент компьютерной инженерии

Курс: Алгоритмизация и программирование

Раздел	**Макс. оценка**	**Итоговая оценка**
Работа программы	1	
Тесты	1	
Правильность алгоритма	3	
Ответы на вопросы	2	
Дополнительное задание	3	
Итого	10	

Отчет по лабораторной работе № 4

Студент: Чапайкин Арсений Георгиевич

Группа: БИВ242

Вариант: № 171 (4 2, 7)

Руководитель: Елисеенко А.М.

Оценка:

Дата сдачи:

МОСКВА 2024

Содержание

Задание

Дано k строк. Каждая строка содержит латинские и русские буквы, цифры, а также все возможные разделители.

1. Выделить из каждой строки и напечатать подстроки, ограниченные с обеих сторон знаками $+$, $-$, $*$, $/$.
2. Среди выделенных подстрок найти самую короткую (если таких подстрок несколько, выбирается первая из них).
3. Преобразовать исходную строку, которой принадлежит найденная подстрока: заменить каждую русскую букву на две такие же буквы.

Листинг программы

```c
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
#include <string.h>

// Reads string to given pointer
size_t getLine(char **string) {
    // Allocate memory for string
    size_t size = 10, index = 0;
    *string = (char *)malloc(sizeof(char) * size);

    // Read string from stdin char by char
    char c = EOF;
    while ((c = getchar()) != EOF && c != '\n') {
        (*string)[index++] = c;
        // Allocate more memory if there is not enough of it
        if (index == size - 1) {
            size *= 2;
            *string = realloc(*string, size);
        }
    }

    // Shrink to fit
    (*string)[index] = '\0';
    *string = realloc(*string, index + 1);

    return index + 1;
}

// Checks if character c is present in substring [left, right] of given string
bool findInSubstr(char c, size_t left, size_t right, char *string) {
    char *substr = (char *)malloc(sizeof(char) * (right - left + 1));
    for (size_t i = left; i <= right; i++) {
        substr[i - left] = string[i];
    }
    char *pos = strchr(substr, c);
    free(substr);
    return pos != NULL;
}

// Checks if given character is delimiter
bool isDelimiter(char c) {
    return c == '+' || c == '-' || c == '*' || c == '/';
}

// Check if given character is cyrillic
bool isCyrillic(int c) {
    char alphabet[] = "йцукенгшщзхъфывапролджэячсмитьбюёЙЦУКЕНГШЩЗХЪФЫВАПРОЛДЖЭЯЧСМИТЬБЮЁ";
    if (strchr(alphabet, c)) {
        return true;
    } else {
        return false;
    }
}

int main() {
    size_t size;
```

```c
    while (scanf("%zu\n", &size) != 1) {
        puts("Wrong input format. Try again");
    }

    // Array of strings
    char **strings = (char **)malloc(sizeof(char *) * size);
    // Length of each string in array
    size_t *length = (size_t *)malloc(sizeof(size_t) * size);

    // Input strings
    for (size_t i = 0; i < size; i++) {
        length[i] = getLine(&strings[i]);
    }

    size_t ind = size + 1;
    size_t begin = size + 2, end = 0;

    puts("\nSubstrings:");
    for (size_t index = 0; index < size; index++) {
        // Iterate through every substring
        for (size_t j = 0; j < length[index]; j++) {
            for (size_t k = j + 1; k < length[index]; k++) {
                // Check if substring is bounded by delimiters
                if (strings[index][j] == strings[index][k] && isDelimiter(strings[
                    index][j])) {
                    // Print substring
                    for (size_t l = j; l <= k; l++) {
                        putchar(strings[index][l]);
                    }
                    putchar('\n');
                    // Update shortest relevant substring
                    if (k - j + 1 < end - begin + 1) {
                        ind = index;
                        begin = j; end = k;
                    }
                }
            }
        }
    }

    // If no relevant substrings were found
    if (ind == size + 1) {
        puts("No relevant substrings found.");
        return 0;
    }

    // Count how many letters are to be doubled
    int cnt_cyrillic = 0;
    for (size_t i = 0; i < length[ind] - 1; i++) {
        char c = strings[ind][i];
        if (isCyrillic(c) && findInSubstr(c, begin, end, strings[ind])) {
            cnt_cyrillic++;
        }
    }

    if (cnt_cyrillic == 0) {
        puts("No cyrillic characters in shortest substring.");
        return 0;
    }

```

```
    // Create new string, doubling each cyrillic character
    char *new_string = (char *)malloc(sizeof(char) * (length[ind] + cnt_cyrillic));
    for (size_t i = 0, pos = 0; i < length[ind] - 1; i++) {
        char c = strings[ind][i];
        if (isCyrillic(c) && findInSubstr(c, begin, end, strings[ind])) {
            new_string[pos++] = c;
        }
        new_string[pos++] = c;
    }

    // Replace old string with new one
    length[ind] += cnt_cyrillic;
    free(strings[ind]);
    strings[ind] = new_string;

    puts("\nResulting string:");
    for (size_t j = 0; j < length[ind] - 1; j++) {
        putchar(strings[ind][j]);
    }
    return 0;
}
```

Распечатка тестов к программе и результатов

Номер	Исходные данные	Результат
1	4 -момо-мо +kiki+wou *jiji ahahaha	Substrings: -момо- +kiki+ Resulting string: -ммоомmоо-ммоо
2	4 -kiki- /анна/ *ахах* jamaika	Substrings: -kiki- /анна/ *ахах* No cyrillic characters in shortest substring.
3	4 -kiki- /анна/ *ахах* - - -	Substrings: -kiki- /анна/ *ахах* - - - - - - - No cyrillic characters in shortest substring.
4	4 opa a podstrok-to net	Substrings: No relevant substrings found.