**Изображение выглядит как текст, небо, внешний

Автоматически созданное описание**

**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ**   
**«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ**   
**імені ІГОРЯ СІКОРСЬКОГО»**

**Інститут прикладного системного аналізу**   
**Кафедра системного проектування**  
   
   
 

**Звіт**

**про виконання лабораторної роботи №12**   
**«****Робота з рядками в С++»**

Виконав:   
студент I курсу, групи ДА-12   
Панасенко Олександр Андрійович

Прийняв:

к.т.н., доцент Безносик О. Ю.

**Мета роботи**

вивчити особливості внутрішнього представлення символьних даних і рядків, навчитися виконувати введення/вивід, перегляд і заміну в рядках при рішенні завдань; вивчити синтаксис і семантику стандартних функцій по роботі з рядками, навчитися використовувати в програмах функції для роботи з рядками на мові C.

**Завдання**

Дано натуральне число n, символи S1, .., Sn. Видалити з S1, .., Sn, усі слова, в яких зустрічається не більше двох різних букв.

**Аналіз умови задачі.**

Виходячи з умови завдання, можна зробити висновок, що для вирішення даної проблеми потрібно написати окрему фунцію, що буде підраховувати кількість унікальних літер у слові. За допомогою змінних типу int я зможу позначати кінець та початок слова у реченні, яке потім зможу копіювати функцією memcpy() у окрему змінну, яку і використаю для підрахунку кількості унікальних літер у слові.

Вхідні дані будуть зчитуватись функцією fgets() та записуватись у масив символів.

**Код програми**

#include <string.h>

#include <malloc.h>

#define FALSE 0

#define TRUE 1

void \_flushout(void)

{

int f = 0;

while ((f = getchar()) != '**\n**' && f != EOF);

}

// checks if a buffer contains a letter

unsigned int \_buffcontains(char\* buffer, size\_t len, char c)

{

for (int i = 0; i < len; i++)

{

if (buffer[i] == c)

return TRUE;

}

return FALSE;

}

// returns number of unique letters in sentence/word

unsigned int \_get\_letters(char\* buffer, size\_t len)

{

unsigned int c = 0;

char\* \_uniquebuff = malloc(len);

for (int i = 0; i < len; i++)

{

if (!\_buffcontains(\_uniquebuff, len, buffer[i]))

{

\_uniquebuff[c] = buffer[i];

c++;

}

}

free(\_uniquebuff);

return c;

}

// deletes words with <= 2 letters

\_\_attribute\_malloc\_\_ char\* \_delete\_odd(const char\* buffer, char delim)

{

size\_t \_act\_buffer\_len = strlen(buffer);

char\* \_newbuff = calloc(\_act\_buffer\_len + 1, sizeof(char)); // <- new buffer without words with less than 2 unique letters

int beg\_point = 0;

int curr\_wlen = 0;

for (size\_t i = 0; i < \_act\_buffer\_len; i++)

{

// debugging purposes

// printf("char %c with id %d\n", buffer[i], buffer[i]);

if (buffer[i] == delim || buffer[i] == '**\n**') // <- check if buffer eq to delim or linefeed, which is a result of user input

{

char\* \_cpy = malloc(curr\_wlen + 1);

memcpy(\_cpy, &buffer[beg\_point], curr\_wlen); // <- will copy a string starting from the beginning of the word til its end

\_cpy[curr\_wlen] = '**\0**';

// this may be commented for testing purposes

unsigned int \_letters = \_get\_letters(\_cpy, curr\_wlen); // <- will return number of unique letters in word

if (\_letters > 2) // <- will remove all the words with less than 2 unique letters

{

strcat(\_newbuff, \_cpy);

strcat(\_newbuff, " ");

}

beg\_point = i + 1; // <- will point on the first letter of the word

curr\_wlen = 0;

free(\_cpy);

}

else curr\_wlen++;

}

return \_newbuff;

}

int main(void)

{

size\_t size;

printf("Enter the size of array:**\n**");

scanf("%d", &size);

\_flushout();

char buffer[size + 1]; // <- extra symbol for \0 that is not added automatically

printf("Enter words:**\n**");

fgets(buffer, sizeof(buffer), stdin);

buffer[size] = '**\0**';

strcat(buffer, " ");

char\* result = \_delete\_odd(buffer, ' ');

printf("Final result:**\n**");

puts(result);

fflush(stdin);

free(result);

}

Результати виконання програми зображено на рисунку 1.

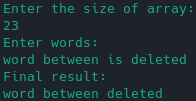


Рисунок 1

**Висновки**

У ході виконання лабораторної роботи №11 я реалізував алгоритм зчитування слів у реченні та підрахунку кількості унікальних букв у цьому слові на мові програмування C. Ці алгоритми разом дозволяють видаляти зайві слова з речення. Протягом виконання лабораторної роботи, я використовув динамічне виділення пам’яті malloc() та free().