Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Алтайский государственный технический университет им. И. И. Ползунова»

Факультет информационных технологий
Кафедра прикладной математики

Отчет защищен с оценкой _____

Преподаватель _____
(подпись)
«___» _____ 2026 г.

Отчет
По лабораторной работе №1

«Работа со строками»

по дисциплине «Программирование – 2 семестр»

Студент группы ПИ-54 Дворников М. С.

Преподаватель доцент,к.т.н. Егорова Е.В.

Барнаул 2025

Задание 1. (№1).
Дан текст. Определить в нем наиболее часто встречающийся символ; затем определить все слова, в которых доля этого символа максимальна. Пример:

Текст: "Veni, scripsi, vixi" ("Пришел, написал, прожил").
Результат: Наиболее часто встречающийся символ - "i". Слова, в которых доля этого символа максимальна: "vixi" (0.5).

Текст файла “mystr.cpp”:
```
char* mystrnset(char* string, const char symbol, int n) {
    // Установить все символы строки string в символы symbol
    // string: изменяемая строка
    // symbol: нужный символ
    // n: сколько символов нужно установить
    for (int i = 0; string[i] && i < n; i++) string[i] = symbol;
    return string;
}
char* mystrchr(char* string, const char symbol) {
    // Найти первое вхождение символа symbol в строке string
    // string: строка, в которой ищем символ
    // symbol: символ, который ищем в строке

    char* point = NULL;
    for (int i = 0; string[i] && !point; i++)
        if (string[i] == symbol) point = string + i;

    return point;
}
int mystrlen(const char* string) {
    // Найти длину указанной строки
    // string: строка, длину которой нужно найти
    int len;
    for (len = 0; string[len]; len++);
    return len;
}
int mystrcount(const char* string, const char symbol) {
    // Функция для подсчёта количества символов symbol в строке string
    // string: строка, в которой считаем символы
    // symbol: символ, количество которого нужно посчитать
    int counter = 0;
    for (int i = 0; string[i]; i++) counter += string[i] == symbol;
    return counter;
}
```

Текст программы (без использования библиотеки <string.h>):

```
#define _CRT_SECURE_NO_WARNINGS

#include <stdio.h>  // Стандартный вход/выход
#include <locale.h>  // Вывод на русском языке
#include <conio.h>  // Задержка экрана

#include <string.h>  // Работа со строками
#include <math.h>  // Математические функции

#include <stdlib.h>  // Создание динамических массивов, использование рандома и преобразования строк
#include <windows.h>  // Очистка консоли и установка кодировки
#include <time.h>  // Обновление рандома


#include "mystr.cpp"  // Своя библиотека для работы со строками


#define RU setlocale(LC_ALL, "Russian")  // Для вывода русских слов
#define UT setlocale(LC_ALL, "ru_RU.UTF-8")  // Для вывода текста файла


#define MAX_LEN 255  // Максимальная длина текста пользователя
#define ALP_LEN 116  // Длина массива для букв
#define ERR 0.1E-5  // Погрешность при вычислениях


#define SEP {' ', ',', '.', '?', '!', '-', ';', ':', '(', ')', '/',  '\\', '|', '\'', '\"', '\0'}  // Символы, которые разделяют слова


int main() {
    SetConsoleCP(1251); SetConsoleOutputCP(1251);  // Ввод на русском
    system("color F0");
    do {
        int free_stream = 0;  // Остались в потоке символы
        // Сначала введём текст и составим алфавит из всех введённых букв
        char sep[] = SEP;  // Символы, которые разделяют слова
        char text[MAX_LEN + 2], alphabet[ALP_LEN + 1];  // Массив с текстом и со всеми встречающимися буквами
        do {
            system("cls");  // Очищаем консоль
            printf("Эта программа определяет в заданном тексте наиболее часто встречающийся символ\n");
            printf("затем определяет все слова, в которых доля этого символа максимальна\n");
            printf("Программа выполнена без использования стандартной библиотеки <string.h>\n\n");

            mystrnset(text, 0, MAX_LEN + 2); mystrnset(alphabet, 0, ALP_LEN + 1);  // Проставляем везде '\0'
            printf("Введите текст (не более %d символов).\nЧтобы закончить ввод нажмите enter:\n", MAX_LEN);
            char letter; int l = 0; int a = 0;
            while ((letter = getchar()) != '\n' && l < MAX_LEN) {
                text[l++] = letter;
                if (!mystrchr(alphabet, letter) && !mystrchr(sep, letter)) alphabet[a++] = letter;
            }
            if (letter == '\n') free_stream = 1;
            text[l] = ' ';  // Ставим знак-разделитель в конце
        } while (!mystrlen(alphabet));  // Повторяем ввод, пока пользователь не введёт хотя бы одну буковку

        /*printf("\nВведённый текст:\n");
        for (int i = 0; text[i]; i++) printf("%c", text[i]);
```

```
        printf("\nВведённый алфавит:\n");
        for (int i = 0; alphabet[i]; i++) printf("%c", alphabet[i]);*/

        // Найдём наиболее часто встречающийся символ
        char often;       // Самый встречающийся символ
        int counter = 0;  // Сколько раз он встречается
        for (int i = 0; alphabet[i]; i++) {
            char cur_letter = alphabet[i];
            int cur_counter = mystrcount(text, cur_letter);
            if (cur_counter > counter) { often = cur_letter; counter = cur_counter; }
        }
        printf("\n\nНаиболее часто встречающийся символ - \"%c\". ", often);

        // Теперь найдём слова, в которых доля этого символа максимальна
        double percent = 0;  // Максимальная доля символа
        char words[MAX_LEN + 2];  // Массив для всех слов, в которых доля этого символа
максимальна
        mystrnset(words, 0, MAX_LEN + 2);  // Проставляем везде '\0'
        int last = 0;  // Индекс последней буквы в массиве words
        int start = 0, end = -1;  // Индексы начала и конца слова;

        for (int i = 0; text[i]; i++) {
            if (mystrchr(alphabet, text[i]))  // Если мы наткнулись на слово, а не разделитель
                if (end == -1) start = end = i;  // Если это начало слова, то ставим и начало
и конец на i
                else end = i;  // Если конец слова уже не равен -1, то обновляем конец слова

            else if (end >= start) { // Если мы наткнулись на разделитель и до этого было
слово (конец слова стоит не перед его началом)

                int cur_counter = 0;  // Сколько в найденном слове нужных букв
                for (int l = start; l <= end; l++)
                    cur_counter += text[l] == often;
                double cur_percent = (double)cur_counter / (end - start + 1);  // Доля символа
в этом слове

                if (fabs(cur_percent - percent) < ERR) {  // Если доли примерно равны
                    for (int l = 0; l <= end - start; l++)  // Запишем в words слово
                        words[last++] = text[start + l];
                    words[last++] = '\"';  // Разделитель между словами
                }
                else if (cur_percent > percent) {  // Если доля больше, то очищаем массив
                    percent = cur_percent;
                    mystrnset(words, 0, last);  // Обновляем массив слов
                    last = 0;  // Обновляем индекс последней буквы
                    for (int l = 0; l <= end - start; l++)  // Запишем в words слово
                        words[last++] = text[start + l];
                    words[last++] = '\"';  // Разделитель между словами
                }
                start = i; end = -1;
            }
        }

        printf("Слова, в которых доля этого символа максимальна: \"");
        for (int i = 0; words[i]; i++)
            if (words[i] == '\"' && words[i + 1]) printf("\", \"");
            else printf("%c", words[i]);
        printf(" (%.2lf)", percent);


        printf("\n\nНажмите ESC, что бы закончить.\nНажмите любую другую кнопку, чтобы
продолжить:\n");
        if (!free_stream) while (getchar() != '\n');  // Очищаем поток
    } while (_getch() != 27);
    return 0;
}
```

Текст программы (с использованием библиотеки <string.h>):

```
#define _CRT_SECURE_NO_WARNINGS

#include <stdio.h>  // Стандартный вход/выход
#include <locale.h>  // Вывод на русском языке
#include <conio.h>  // Задержка экрана

#include <string.h>  // Работа со строками
#include <math.h>  // Математические функции

#include <stdlib.h>  // Создание динамических массивов, использование рандома и преобразования строк
#include <windows.h>  // Очистка консоли и установка кодировки
#include <time.h>  // Обновление рандома


#include "mystr.cpp"  // Своя библиотека для работы со строками


#define RU setlocale(LC_ALL, "Russian")  // Для вывода русских слов
#define UT setlocale(LC_ALL, "ru_RU.UTF-8")  // Для вывода текста файла


#define MAX_LEN 255  // Максимальная длина текста пользователя
#define ALP_LEN 116  // Длина массива для букв
#define ERR 0.1E-5  // Погрешность при вычислениях


int main() {
    SetConsoleCP(1251); SetConsoleOutputCP(1251);  // Ввод на русском
    system("color F0");
    const char* sep = " ,.?!-;:()/\\|\"\'\n\0";
    do {
        // Сначала введём текст и преобразуем его в массив слов
        char* user_input, * text_string;  // Строка для ввода пользователя и для хранения текста в виде строчки
        user_input = (char*)calloc(MAX_LEN + 1, sizeof(char)); text_string = (char*)calloc(MAX_LEN + 1, sizeof(char));
        char* text_arr[MAX_LEN / 2];  // Массив для слов
        do {
            system("cls");  // Очищаем консоль
            printf("Эта программа определяет в заданном тексте наиболее часто встречающийся символ\n");
            printf("затем определяет все слова, в которых доля этого символа максимальна\n");
            printf("Программа выполнена с использованием стандартной библиотеки <string.h>\n\n");

            printf("Введите текст (не более %d символов).\nЧтобы закончить ввод нажмите enter:\n", MAX_LEN);
            _strnset(user_input, 0, MAX_LEN + 1); _strnset(text_string, 0, MAX_LEN + 1);  // Проставляем везде '\0'
            fgets(user_input, MAX_LEN + 1, stdin);  // Ввод пользователя
            strncpy(text_string, user_input, MAX_LEN + 1);  // Копируем в text_string
            text_arr[0] = strtok(user_input, sep);
        } while (!text_arr[0]);
        for (int i = 1; text_arr[i] = strtok(NULL, sep); i++);


        /*printf("\nПолучился такой массив из слов:\n\"%s\"", text_arr[0]);
        for (int i = 1; text_arr[i]; i++) printf(", \"%s\"", text_arr[i]);
        printf("\n");*/
```

```
        // Найдём наиболее часто встречающийся символ
        char often = 0;       // Самый встречающийся символ
        int counter = 0;  // Сколько раз он встречается
        for (int i = 0; text_arr[i]; i++)
            for (int j = 0; text_arr[i][j]; j++) {
                char cur_letter = text_arr[i][j];
                int cur_counter = mystrcount(text_string, cur_letter);
                if (cur_counter > counter) { often = cur_letter; counter = cur_counter; }
            }
        printf("\n\nНаиболее часто встречающийся символ - \"%c\". ", often);

        // Теперь найдём слова, в которых доля этого символа максимальна
        double percent = 0;  // Максимальная доля символа
        char* words[MAX_LEN / 2];  // Массив для всех слов, в которых доля этого символа
максимальна
        int last = 0;  // Индекс последнего слова в массиве words

        for (int i = 0; text_arr[i]; i++) {
            int cur_counter = mystrcount(text_arr[i], often);
            double cur_percent = (double)cur_counter / strlen(text_arr[i]);  // Доля символа в
этом слове

            if (fabs(cur_percent - percent) < ERR)
                words[last++] = text_arr[i];

            else if (cur_percent > percent) {
                percent = cur_percent;
                for (int w = 0; w < last; w++) words[w] = 0;
                last = 0;
                words[last++] = text_arr[i];
            }
        }

        printf("Слова, в которых доля этого символа максимальна: ");
        printf("\"%s\"", words[0]);
        for (int w = 1; w < last; w++) printf(", \"%s\"", words[w]);
        printf(" (%.2lf)", percent);

        printf("\n\nНажмите ESC, что бы закончить.\nНажмите любую другую кнопку, чтобы
продолжить:\n");
        if (text_string[MAX_LEN - 1]) while (getchar() != '\n');  // Очищаем поток
        free(text_string);
    } while (_getch() != 27);
    return 0;
}
```

Тесты программы:
Так как обе программы выполняют одну и ту же задачу, их входные и выходные данные должны совпадать. Поэтому тестироваться будут две программы, но отчёт будет содержать скриншот тестов только одной (так как в ином случае будет два одинаковых скриншота).
Сначала немного поменяем код и поставим максимальную длину ввода на 16 и проверим крайние значения:
Ввод enter первым же символом запускает всё заново. Так же получается при вводе строки, состоящей только из знаков-разделителей:

```
Эта программа определяет в заданном тексте наиболее часто встречающийся символ
затем определяет все слова, в которых доля этого символа максимальна
Программа выполнена с использованием стандартной библиотеки <string.h>

Введите текст (не более 16 символов).
Чтобы закончить ввод нажмите enter:
```

Ввод строки, длина которой превышает заданное число:

```
Введите текст (не более 16 символов).
Чтобы закончить ввод нажмите enter:
лллллллллллллллллллллллллллллллллллллллллллллллллл


Наиболее часто встречающийся символ - "л". Слова, в которых доля этого символа максимальна: "ллллллллллл
ллллл" (1.00)

Нажмите ESC, что бы закончить.
Нажмите любую другую кнопку, чтобы продолжить:
```

То есть строка обрезается

Ввод одного слова или одной буквы ничем не отличается от обычной работы программы:

```
Введите текст (не более 16 символов).
Чтобы закончить ввод нажмите enter:
Приветики


Наиболее часто встречающийся символ - "и". Слова, в которых доля этого символа максимальна: "Приветики"
(0.33)

Нажмите ESC, что бы закончить.
Нажмите любую другую кнопку, чтобы продолжить:
Введите текст (не более 16 символов).
Чтобы закончить ввод нажмите enter:
ь


Наиболее часто встречающийся символ - "ь". Слова, в которых доля этого символа максимальна: "ь" (1.00)
```

Далее проведём пару тестов из примеров (поставим максимальную длину 255):

```
Эта программа определяет в заданном тексте наиболее часто встречающийся символ
затем определяет все слова, в которых доля этого символа максимальна
Программа выполнена без использования стандартной библиотеки <string.h>

Введите текст (не более 255 символов).
Чтобы закончить ввод нажмите enter:
Veni, scripsi, vixi


Наиболее часто встречающийся символ - "i". Слова, в которых доля этого символа максимальна: "vixi" (0.50)

Нажмите ESC, что бы закончить.
Нажмите любую другую кнопку, чтобы продолжить:
Эта программа определяет в заданном тексте наиболее часто встречающийся символ
затем определяет все слова, в которых доля этого символа максимальна
Программа выполнена с использованием стандартной библиотеки <string.h>

Введите текст (не более 255 символов).
Чтобы закончить ввод нажмите enter:
Veni, vidi, vici


Наиболее часто встречающийся символ - "i". Слова, в которых доля этого символа максимальна: "vidi", "vici" (0.50)

Нажмите ESC, что бы закончить.
Нажмите любую другую кнопку, чтобы продолжить:
```

Задание 2. (№55)

Дан текст и отдельное слово. Распечатать те слова текста, которые можно составить из букв, входящих в заданное слово. Каждую букву можно использовать не более числа ее вхождений в заданное слово. Пример:
Исходный текст: «Сухая галета лежала на газете». ***Слово***: «телегазета».
Результат: Слова: галета газете.

Текст программы (без использования библиотеки <string.h>):

```
#define _CRT_SECURE_NO_WARNINGS

#include <stdio.h>  // Стандартный вход/выход
#include <locale.h>  // Вывод на русском языке
#include <conio.h>  // Задержка экрана

#include <string.h>  // Работа со строками
#include <math.h>  // Математические функции

#include <stdlib.h>  // Создание динамических массивов, использование рандома и преобразования строк
#include <windows.h>  // Очистка консоли и установка кодировки
#include <time.h>  // Обновление рандома


#include "mystr.cpp"  // Своя библиотека для работы со строками


#define RU setlocale(LC_ALL, "Russian")  // Для вывода русских слов
#define UT setlocale(LC_ALL, "ru_RU.UTF-8")  // Для вывода текста файла


#define MAX_LEN 255  // Максимальная длина текста пользователя


#define SEP {' ', ',', '.', '?', '!', '-', ';', ':', '(', ')', '/',  '\\', '|', '\'', '\"', '\n', '\0'}  // Символы, которые разделяют слова
```

```
int main() {
    SetConsoleCP(1251); SetConsoleOutputCP(1251);  // Ввод на русском
    system("color F0");
    do {  // Ввод текста
        char sep[] = SEP;  // Символы, которые разделяют слова
        char text[MAX_LEN + 2];  // Массив с текстом
        char word[MAX_LEN / 2 + 1];  // Массив со словом
        int is_letter;  // Флаг "хотя бы раз встретилась буква"
        do {
            system("cls");  // Очищаем консоль
            printf("Эта программа печатает те слова заданного текста, которые можно\n");
            printf("составить из букв, входящих в заданное слово.Каждую букву можно
использовать не\n");
            printf("более числа ее вхождений в заданное слово\n");
            printf("Программа выполнена без использования стандартной библиотеки
<string.h>\n\n");

            mystrnset(text, 0, MAX_LEN + 2);  // Проставляем везде '\0'
            printf("Введите текст (не более %d символов).\nЧтобы закончить ввод нажмите
enter:\n", MAX_LEN);
            char letter; int l = 0;
            is_letter = 0;
            while ((letter = getchar()) != '\n' && l < MAX_LEN) {
                text[l++] = letter;
                if (!mystrchr(sep, letter)) is_letter = 1;
            }
            if (letter != '\n') while (getchar() != '\n');  // Очищаем поток
            text[l] = ' ';  // Ставим знак-разделитель в конце
        } while (!is_letter);  // Повторяем ввод, пока пользователь не введёт хотя бы одну
буковку

        do {  // Ввод слова
            system("cls");  // Очищаем консоль
            printf("Эта программа печатает те слова заданного текста, которые можно\n");
            printf("составить из букв, входящих в заданное слово.Каждую букву можно
использовать не\n");
            printf("более числа ее вхождений в заданное слово\n");
            printf("Программа выполнена без использования стандартной библиотеки
<string.h>\n\n");

            printf("\nВведённый текст:\n");
            for (int i = 0; text[i]; i++) printf("%c", text[i]);

            mystrnset(word, 0, MAX_LEN / 2 + 1);  // Проставляем везде '\0
            printf("\n\nВведите слово (не более %d символов).\nЧтобы закончить ввод нажмите
enter:\n", MAX_LEN / 2);
            char letter; int l = 0;
            is_letter = 0;
            while (!mystrchr(sep, letter = getchar()) && l < MAX_LEN / 2) word[l++] = letter;
            if (letter != '\n') while (getchar() != '\n');  // Очищаем поток
        } while (!mystrlen(word));  // Повторяем ввод, пока пользователь не введёт хотя бы
одну буковку
        mystrlwr(word);

        // Теперь найдём слова, которые можно составить из букв, входящих в заданное слово
        char words[MAX_LEN + 2];  // Массив для слов
        mystrnset(words, 0, MAX_LEN + 2);  // Проставляем везде '\0'
        int last = 0;  // Индекс последней буквы в массиве words
        int start = 0, end = -1;  // Индексы начала и конца слова;

        for (int i = 0; text[i]; i++) {
            if (!mystrchr(sep, text[i]))  // Если мы наткнулись на слово, а не разделитель
                if (end == -1) start = end = i;  // Если это начало слова, то ставим и начало
и конец на i
                else end = i;  // Если конец слова уже не равен -1, то обновляем конец слова
```

```
            else if (end >= start) { // Если мы наткнулись на разделитель и до этого было
слово (конец слова стоит не перед его началом
                int is_good = end - start + 1 <= MAX_LEN / 2;  // Подходит ли нам слово
                char cur_word[MAX_LEN / 2 + 2];  // Текущее слово
                int letter;
                for (letter = 0; letter <= end - start && letter < MAX_LEN / 2; letter++)
cur_word[letter] = text[start + letter];
                cur_word[letter++] = '\"'; cur_word[letter] = '\0';

                char lower_cur_word[MAX_LEN / 2 + 2];
                mystrcpy(lower_cur_word, cur_word);
                mystrlwr(lower_cur_word);

                for (int j = 0; j < letter - 1 && is_good; j++)
                    if (mystrcount(lower_cur_word, lower_cur_word[j]) > mystrcount(word,
lower_cur_word[j])) is_good = 0;
                if (is_good) mystrcat(words, cur_word);
                start = i; end = -1;
            }
        }

        if (mystrlen(words)) {
            printf("Слова: \"");
            for (int i = 0; words[i]; i++)
                if (words[i] == '\"' && words[i + 1]) printf("\", \"");
                else printf("%c", words[i]);
        }
        else {
            printf("Нет слов, которые можно составить из букв \"");
            for (int i = 0; word[i]; i++) printf("%c", word[i]);
            printf("\"\n");
        }

        printf("\n\nНажмите ESC, что бы закончить.\nНажмите любую другую кнопку, чтобы
продолжить:\n");
    } while (_getch() != 27);
    return 0;
}
```

Текст программы (с использованием библиотеки <string.h>):

```
#define _CRT_SECURE_NO_WARNINGS

#include <stdio.h>  // Стандартный вход/выход
#include <locale.h>  // Вывод на русском языке
#include <conio.h>  // Задержка экрана

#include <string.h>  // Работа со строками
#include <math.h>  // Математические функции

#include <stdlib.h>  // Создание динамических массивов, использование рандома и преобразования строк
#include <windows.h>  // Очистка консоли и установка кодировки
#include <time.h>  // Обновление рандома


#include "mystr.cpp"  // Своя библиотека для работы со строками


#define RU setlocale(LC_ALL, "Russian")  // Для вывода русских слов
#define UT setlocale(LC_ALL, "ru_RU.UTF-8")  // Для вывода текста файла


#define MAX_LEN 255  // Максимальная длина текста пользователя


int main() {
    SetConsoleCP(1251); SetConsoleOutputCP(1251);  // Ввод на русском
    system("color F0");
    const char* sep = " ,.?!-;:()/\\|\"\'\n\0";
    do {  // Ввод текста
        char* user_input;  // Строка для ввода пользователя и для хранения текста в виде строчки
        user_input = (char*)calloc(MAX_LEN + 1, sizeof(char));
        char* text_arr[MAX_LEN / 2];  // Массив для слов
        do {
            system("cls");  // Очищаем консоль
            printf("Эта программа печатает те слова заданного текста, которые можно\n");
            printf("составить из букв, входящих в заданное слово.Каждую букву можно использовать не\n");
            printf("более числа ее вхождений в заданное слово\n");
            printf("Программа выполнена с использованием стандартной библиотеки <string.h>\n\n");

            printf("Введите текст (не более %d символов).\nЧтобы закончить ввод нажмите enter:\n", MAX_LEN);
            _strnset(user_input, 0, MAX_LEN + 1);  // Проставляем везде '\0'
            fgets(user_input, MAX_LEN + 1, stdin);  // Ввод пользователя
            if (user_input[MAX_LEN - 1]) while (getchar() != '\n');  // Очищаем поток
            text_arr[0] = strtok(user_input, sep);
        } while (!text_arr[0]);
        for (int i = 1; text_arr[i] = strtok(NULL, sep); i++);


        char* word;  // Строка-слово
        int good_word;  // Пользователь успешно ввёл слово
        do {  // Ввод слова
            good_word = 1;
            system("cls");  // Очищаем консоль
            printf("Эта программа печатает те слова заданного текста, которые можно\n");
            printf("составить из букв, входящих в заданное слово.Каждую букву можно использовать не\n");
            printf("более числа ее вхождений в заданное слово\n");
            printf("Программа выполнена без использования стандартной библиотеки <string.h>\n\n");
```

```
            printf("\nВведённые слова:\n");
            printf("\"%s\"", text_arr[0]);
            for (int i = 1; text_arr[i]; i++) printf(", \"%s\"", text_arr[i]);

            word = (char*)calloc(MAX_LEN / 2 + 1, sizeof(char));
            printf("\n\nВведите слово (не более %d символов).\nЧтобы закончить ввод нажмите
enter:\n", MAX_LEN / 2);
            fgets(word, MAX_LEN / 2 + 1, stdin);
            if (word[MAX_LEN / 2 - 1]) while (getchar() != '\n');  // Очищаем поток

            if (!word[0]) good_word = 0;
            else if (!(word = strtok(word, sep))) good_word = 0;
        } while (!good_word);  // Повторяем ввод, пока пользователь не введёт хотя бы одну
буковку
        _strlwr(word);

        // Теперь найдём слова, которые можно составить из букв, входящих в заданное слово
        int second_word = 0;  // Мы нашли хотя бы одно слово
        for (int w = 0; text_arr[w]; w++)
            if (strlen(text_arr[w]) <= strlen(word)) {
                int is_good = 1;  // Подходит ли нам слово

                char* lower_cur_word;
                lower_cur_word = (char*)calloc(MAX_LEN / 2 + 1, sizeof(char));
                strcpy(lower_cur_word, text_arr[w]);
                _strlwr(lower_cur_word);

                for (int i = 0; lower_cur_word[i] && is_good; i++)
                    if (mystrcount(lower_cur_word, lower_cur_word[i]) > mystrcount(word,
lower_cur_word[i]))
                        is_good = 0;

                if (is_good) {
                    if (second_word) printf(", \"%s\"", text_arr[w]);
                    else { second_word = 1; printf("Слова: \"%s\"", text_arr[w]); }
                }
                free(lower_cur_word);
            }

        if (!second_word) printf("Нет слов, которые можно составить из букв \"%s\"", word);

        printf("\n\nНажмите ESC, что бы закончить.\nНажмите любую другую кнопку, чтобы
продолжить:\n");
    } while (_getch() != 27);
    return 0;
}
```

Тесты программы:
Так как обе программы выполняют одну и ту же задачу, их входные и выходные данные должны совпадать. Поэтому тестироваться будут две программы, но отчёт будет содержать скриншот тестов только одной (так как в ином случае будет два одинаковых скриншота).
Сначала немного поменяем код и поставим максимальную длину ввода на 16 и проверим крайние значения:
Ввод enter первым же символом запускает всё заново. Так же получается при вводе строки, состоящей только из знаков-разделителей. Это работает как для ввода текста, так и для ввода слова:

```
Эта программа печатает те слова заданного текста, которые можно
составить из букв, входящих в заданное слово.Каждую букву можно использовать не
более числа ее вхождений в заданное слово
Программа выполнена с использованием стандартной библиотеки <string.h>

Введите текст (не более 16 символов).
Чтобы закончить ввод нажмите enter:
```

```
Эта программа печатает те слова заданного текста, которые можно
составить из букв, входящих в заданное слово.Каждую букву можно использовать не
более числа ее вхождений в заданное слово
Программа выполнена без использования стандартной библиотеки <string.h>


Введённые слова:
"вывы"

Введите слово (не более 8 символов).
Чтобы закончить ввод нажмите enter:
```

Ввод строки, длина которой превышает заданное число:

```
Введённые слова:
"вввввввввввввввв"

Введите слово (не более 8 символов).
Чтобы закончить ввод нажмите enter:
вввввввввввввввввввввввв
Нет слов, которые можно составить из букв "ввввввв"

Нажмите ESC, что бы закончить.
Нажмите любую другую кнопку, чтобы продолжить:
```

Ввод одного слова или одной буквы ничем не отличается от обычной работы программы:

```
Введённые слова:
"а"

Введите слово (не более 8 символов).
Чтобы закончить ввод нажмите enter:
а
Слова: "а"
```

Далее проведём пару тестов из примеров (поставим максимальную длину 255):

```
Эта программа печатает те слова заданного текста, которые можно
составить из букв, входящих в заданное слово.Каждую букву можно использовать не
более числа ее вхождений в заданное слово
Программа выполнена без использования стандартной библиотеки <string.h>


Введённый текст:
Сухая галета лежала на газете

Введите слово (не более 127 символов).
Чтобы закончить ввод нажмите enter:
телегазета
Слова: "галета", "газете"

Нажмите ESC, что бы закончить.
Нажмите любую другую кнопку, чтобы продолжить:
```

```
Эта программа печатает те слова заданного текста, которые можно
составить из букв, входящих в заданное слово.Каждую букву можно использовать не
более числа ее вхождений в заданное слово
Программа выполнена без использования стандартной библиотеки <string.h>


Введённые слова:
"Сухая", "галета", "лежала", "на", "газете"

Введите слово (не более 127 символов).
Чтобы закончить ввод нажмите enter:
телегазета
Слова: "галета", "газете"

Нажмите ESC, что бы закончить.
Нажмите любую другую кнопку, чтобы продолжить:
```