1. Министерство образования и науки Российской Федерации
2. Санкт-Петербургский Политехнический Университет Петра Великого
3. —
4. Институт кибербезопасности и защиты информации
5. **Кафедра «Информационная безопасность компьютерных систем»**

**ЛАБОРАТОРНАЯ РАБОТА № 1**

1. «**Работа с файлами**»
2. по дисциплине «Структуры данных»
3. Выполнил
4. студенты гр. 5131001/40003 Веденеев А. С., Поляков Д. Т.

<*подпись*>

1. Преподаватель
2. Вагисаров В. Б.

<*подпись*>

1. Санкт-Петербург
2. 2025

Оглавление

[1 Цель работы 3](#_Toc193722231)

[2 Постановка задачи 3](#_Toc193722232)

[3 Описание решения 3](#_Toc193722233)

[4 Тестирование и результаты работы программы 3](#_Toc193722234)

[5 Выводы 3](#_Toc193722235)

[6 Приложения 3](#_Toc193722236)

# Цель работы

Изучение работы с файлами в языке программирования C, освоение методов чтения и записи данных, а также практическая реализация алгоритма замены слов в текстовом файле с использованием динамического выделения памяти и обработки аргументов командной строки.

# Постановка задачи

Разработать программу на языке C, которая выполняет замену заданных слов в тексте, считывая исходные данные из файлов. Программа должна корректно обрабатывать текст, сохранять структуру документа, эффективно управлять памятью и получать пути к файлам через аргументы командной строки.

# Описание решения

Возможные подходы к решению:

1. Построчная обработка текста

* Считывание текста из выходного файла построчно.
* Разбиение строки на слова и замена необходимых слов.
* Запись измененной строки в выходной файл.
* Плюсы: простая реализация, меньший расход памяти.
* Минусы: может не учитывать переносы строк внутри текста или сложные структуры.

1. Посимвольная обработка текста (выбранный метод)
   * Чтение входного файла символ за символом.
   * Построение слов из последовательности символов.
   * Поиск заменяемых слов и запись результата в выходной файл.
   * Плюсы: точный контроль за символами, возможность сохранить оригинальную структуру текста (включая пробелы, знаки препинания и спецсимволы).
   * Минусы: чуть более сложная реализация, требует управления динамической памятью.

Обоснование выбора:

Для данной задачи важна точность замены слов без изменения структуры текста. Подход с построчной обработкой мог бы исказить оригинальное форматирование, тогда как посимвольный анализ позволяет сохранить исходную расстановку знаков препинания, пробелов и специальных символов. Этот метод также обеспечивает эффективное управление памятью, так как не требует загрузки всего файла в оперативную память.

# Тестирование и результаты работы программы

|  |  |
| --- | --- |
| Входные данные: | Результат: |
| ./main.exe input.txt old\_words.txt new\_words.txt output.txt | Output.txt |

Примеры работы программы:

**Версия 1.**

Файл 1 (input.txt) – Исходный текст

**Alice** was beginning to get very tired of sitting by her sister on the bank, and of having nothing to do.

Once or twice she had peeped into the **book** her sister was reading, but it had no **pictures** or conversations in it.

"And what is the use of a **book**," thought **Alice**, "without **pictures** or **conversation**?"

Файл 2 (old\_words.txt) – Слова для замены

Alice

book

pictures

conversation

Файл 3 (new\_words.txt) – Слова-замены

Lena

novel

illustrations

discussion

Ожидаемые выходные данные (файл 4 – output.txt)

Lena was beginning to get very tired of sitting by her sister on the bank, and of having nothing to do.

Once or twice she had peeped into the novel her sister was reading, but it had no illustrations or conversations in it.

"And what is the use of a novel," thought Lena, "without illustrations or discussion?"

**Версия 2. Дополнительные случаи**

Файл 1 (input.txt) – Исходный текст

Hello, world! Alice's adventure begins...

"Look at this book!" she said. "It has many pictures and interesting conversation."

"And what is the use of a book," thought Alice, "without picturesor conversation?"

Файл 2 (old\_words.txt) – Слова для замены

Hello

Alice

book

pictures

conversation

Файл 3 (new\_words.txt) – Слова-замены

Hi

Emma

magazine

illustrations

discussion

Ожидаемые выходные данные (файл 4 – output.txt)

Hi, world! Emma's adventure begins...

"Look at this magazine!" she said. "It has many illustrations and interesting discussion."

# Выводы

В ходе выполнения лабораторной работы были изучены принципы работы с текстовыми файлами, включая их чтение, запись и обработку. Особое внимание было уделено обработке файлов большого объема и управлению памятью при динамическом выделении массивов.

Также были изучены методы поиска и замены подстрок в строках, а именно работа с функциями strcmp, strcpy и динамическое расширение буфера. В процессе выполнения работы были выявлены и исправлены ошибки, связанные с утечками памяти, некорректной обработкой границ слов и пропуском последнего слова в файле.

Благодаря этому опыту удалось значительно улучшить навыки работы с языком C, в частности, управление памятью, работу с динамическими структурами данных и обработку текстовых данных в потоках ввода-вывода.

# Приложения

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#define MAX\_WORDS 300

#define INITIAL\_BUFFER\_SIZE 200

// функция для чтения слов из файла с динамическим выделением памяти

int read\_words(FILE\* file, char\*\*\* words) {

    size\_t count = 0;

    \*words = malloc(sizeof(char\*) \* MAX\_WORDS); // выделяет память под массив из MAX\_WORDS указателей на строки.

    if (\*words == NULL) {

        perror("memory allocation failed");

        exit(EXIT\_FAILURE);

    }

    char buffer[INITIAL\_BUFFER\_SIZE];

    while (fgets(buffer, INITIAL\_BUFFER\_SIZE, file) && count < MAX\_WORDS) {

        size\_t len = strlen(buffer);

        // Убираем лишние пробельные символы

        while (len > 0 && (buffer[len - 1] == '\r' || buffer[len - 1] == '\n' || buffer[len - 1] == ' ')) {

            buffer[--len] = '\0';

        }

        (\*words)[count] = malloc(len + 1); // выделяем память под строку, учитывая \0 - len + 1

        if ((\*words)[count] == NULL) {

            perror("memory allocation failed");

            exit(EXIT\_FAILURE);

        }

        strcpy((\*words)[count], buffer); // копирование строки в выделенную память

        count++;

    }

    return count;

}

int main(int argc, char\* argv[]) {

    if (argc != 5) {

        fprintf(stderr, "Usage: %s <input\_file> <replace\_words\_file> <replacement\_words\_file> <output\_file>\n", argv[0]);

        return EXIT\_FAILURE;

    }

    FILE\* input\_file = fopen(argv[1], "r");

    FILE\* words\_file = fopen(argv[2], "r");

    FILE\* replace\_file = fopen(argv[3], "r");

    FILE\* output\_file = fopen(argv[4], "w");

    if (!input\_file || !words\_file || !replace\_file || !output\_file) {

        perror("[error opening file]");

        return EXIT\_FAILURE;

    }

    // указатель на указатель -> двумерный массив

    char\*\* words\_to\_replace;

    char\*\* replacement\_words;

    int replace\_count = read\_words(words\_file, &words\_to\_replace); // &words\_to\_replace - char\*\*\*

    int replacement\_count = read\_words(replace\_file, &replacement\_words);

    if (replace\_count != replacement\_count) {

        fprintf(stderr, "Error: Mismatched replace and replacement word counts\n");

        return EXIT\_FAILURE;

    }

    int ch;

    char\* buffer = malloc(INITIAL\_BUFFER\_SIZE); // считываемое слово

    size\_t buf\_size = INITIAL\_BUFFER\_SIZE;

    size\_t index = 0; // текущая позиция с считываемом слове

    if (buffer == NULL) {

        perror("memory allocation failed");

        return EXIT\_FAILURE;

    }

    // считывание каждого символа до конца файла

    while ((ch = fgetc(input\_file)) != EOF) {

        // построение слова из символов

        if ((ch >= 'A' && ch <= 'Z') || (ch >= 'a' && ch <= 'z') || ch == '-' || ch == '\'') {

            //

            if (index >= buf\_size - 1) {

                buf\_size \*= 2;

                buffer = realloc(buffer, buf\_size);

                if (buffer == NULL) {

                    perror("memory allocation failed");

                    return EXIT\_FAILURE;

                }

            }

            buffer[index++] = ch;

            buffer[index] = '\0';

        // окончание слова

        } else {

            if (index > 0) { // проверяем, есть ли слово в буфере

                int replaced = 0;

                for (int i = 0; i < replace\_count; i++) {

                    if (strcmp(buffer, words\_to\_replace[i]) == 0) {

                        fprintf(output\_file, "%s", replacement\_words[i]);

                        replaced = 1;

                        break;

                    }

                }

                if (!replaced) {

                    fprintf(output\_file, "%s", buffer);

                }

                index = 0;

            }

            fputc(ch, output\_file);

        }

    }

    free(buffer);

    for (int i = 0; i < replace\_count; i++) {

        free(words\_to\_replace[i]);

        free(replacement\_words[i]);

    }

    free(words\_to\_replace);

    free(replacement\_words);

    fclose(input\_file);

    fclose(words\_file);

    fclose(replace\_file);

    fclose(output\_file);

    printf("[the operation was successful]\n");

    return EXIT\_SUCCESS;

}