ФИЛИАЛ ФЕДЕРАЛЬНОГО ГОСУДАРСТВЕННОГО БЮДЖЕТНОГО
ОБРАЗОВАТЕЛЬНОГО УЧРЕЖДЕНИЯ ВЫСШЕГО ОБРАЗОВАНИЯ
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ «МЭИ»
В Г. СМОЛЕНСКЕ

Кафедра вычислительной техники

Дисциплина: ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ
АВТОМАТИЗИРОВАННЫХ СИСТЕМ

Лабораторная работа № 8.
ОСНОВЫ ПРОГРАММИРОВАНИЯ В LINUX

Студент: Старостенков А.А.
Группа: ВМ-22 (маг.)
Вариант: 15
Преподаватель: Федулов Я.А.

СМОЛЕНСК
2023

Задание:

Создайте файл исходного текста программы с помощью редактора nano.

a) Разработайте алгоритм программы в соответствии с вариантом задания.
b) Разбейте алгоритм на несколько логических блоков.
c) Создайте исходный текст программы, реализующей разработанный алгоритм.
d) Скомпилируйте проект командой gcc program.c – program, где program — название программы.
e) Устраните синтаксические ошибки, если такие имеются.
f) Разработайте набор тестов с учетом особенностей реализации алгоритма.
g) Проверьте правильность функционирования программы с использованием набора тестов.
h) Устраните семантические ошибки в программе, если такие имеются.

Вариант 15.
Написать программу подсчета количества слов для всех текстовых файлов *.txt заданного каталога. Формат вызова программы: wordcount <путь>. Программа должна выводить список файлов в два столбца: имя файла, количество слов в файле.

Выполнение

Для удобства напишем bash-скрипт генерации и заполнения текстовых файлов в указанном каталоге. Так мы облегчим себе работу, ведь не придётся самостоятельно создавать текстовые файлы, заполнять их словами и т.д. Мы автоматизируем этот процесс.

Этот скрипт будет принимать два основных параметра: путь к каталогу и количество файлов для создания.

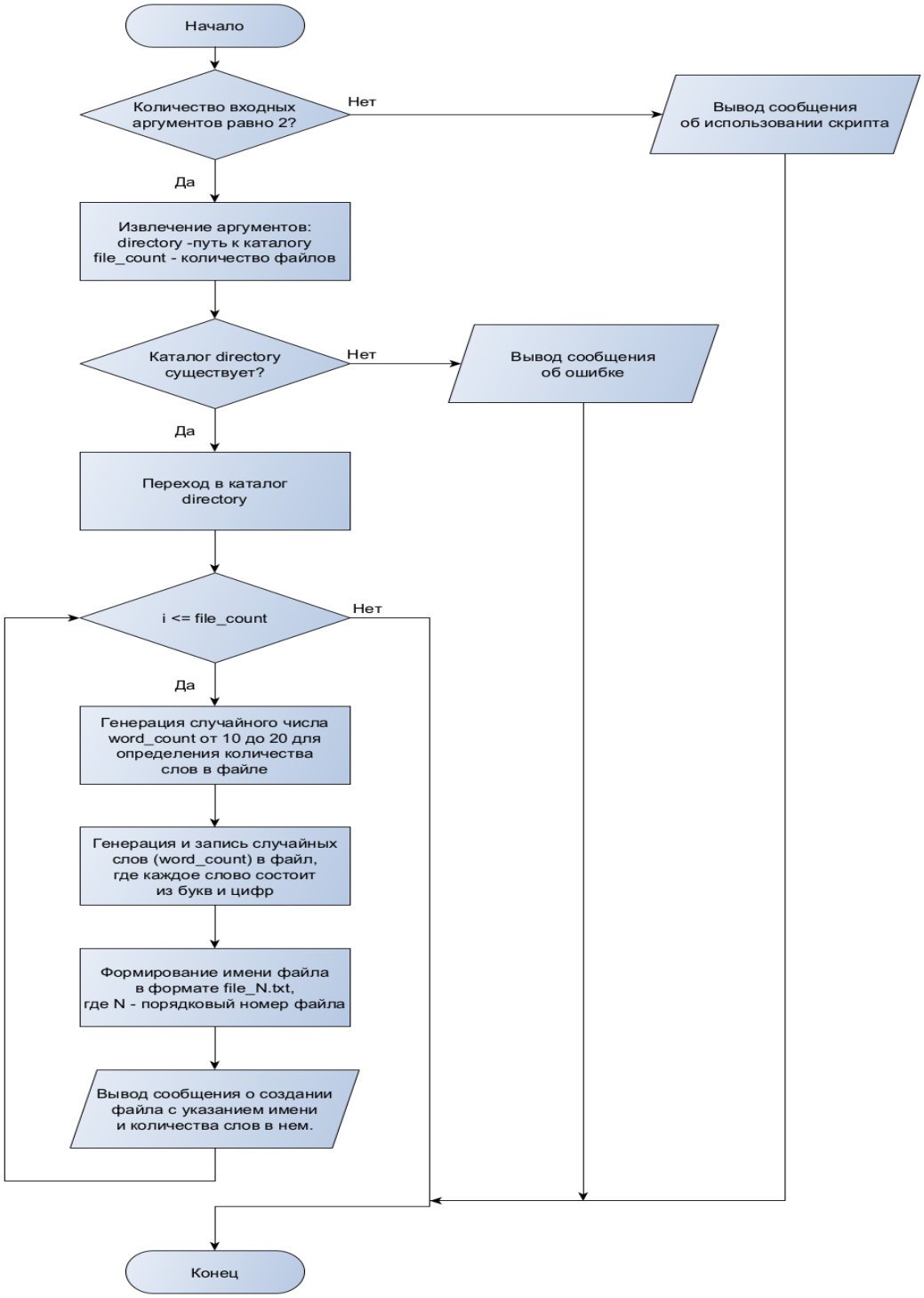

Рисунок 1 – схема алгоритма генерации текстовых файлов

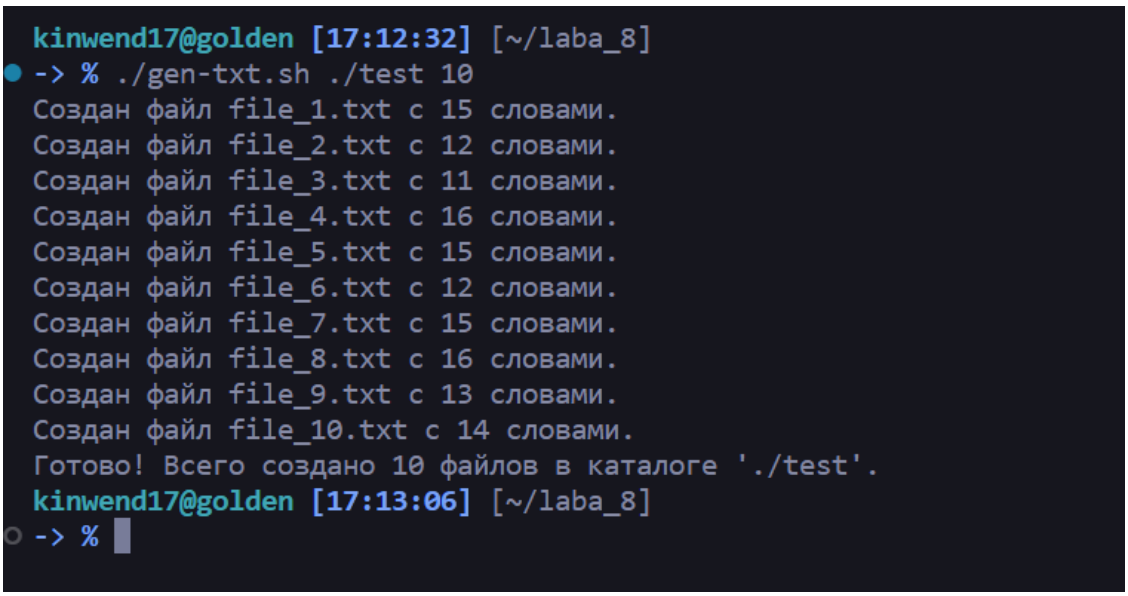

Рисунок 2 – Пример вывода bash-скрипта генерации файлов

Наша задача заключается в написании программы на Си, который выполняет подсчет количества слов во всех текстовых файлах (*.txt) в указанном каталоге и выводит результат в формате «имя файла – количество слов». Вызов программы должен быть в формате wordcount <путь>.

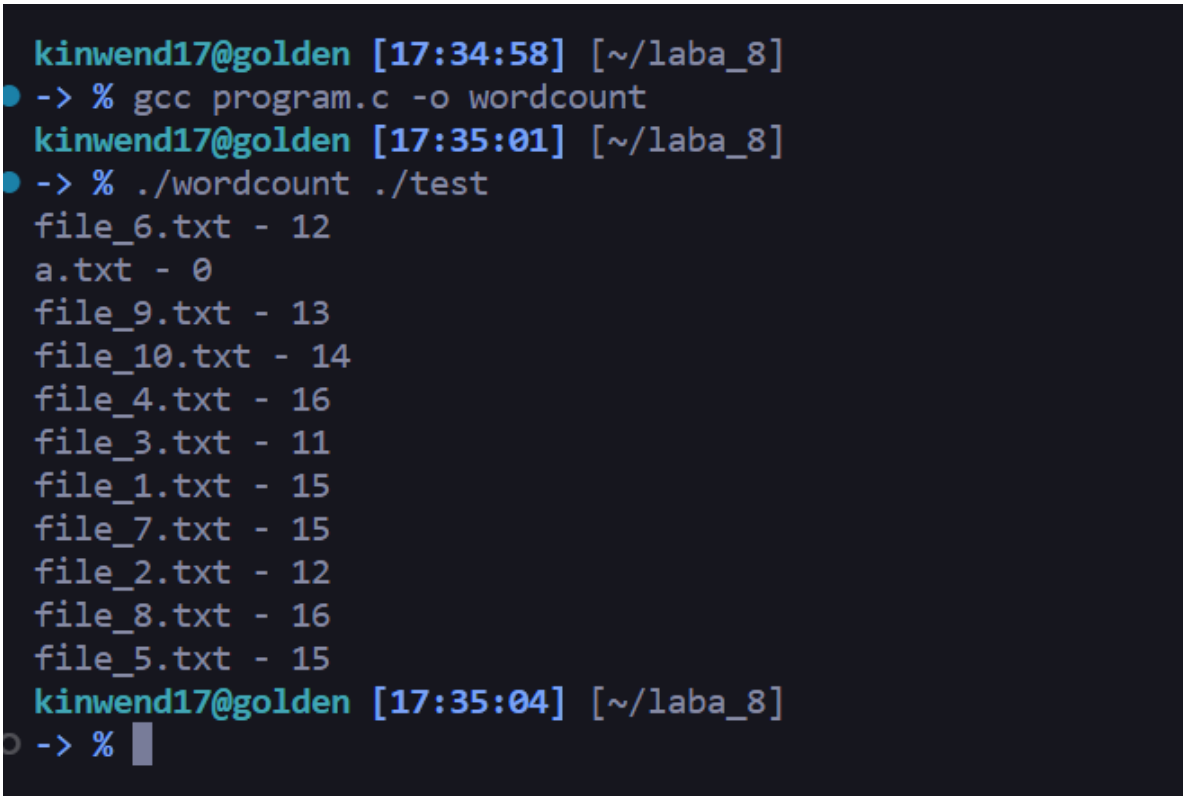

Рисунок 3 – Вывод разработанной программы

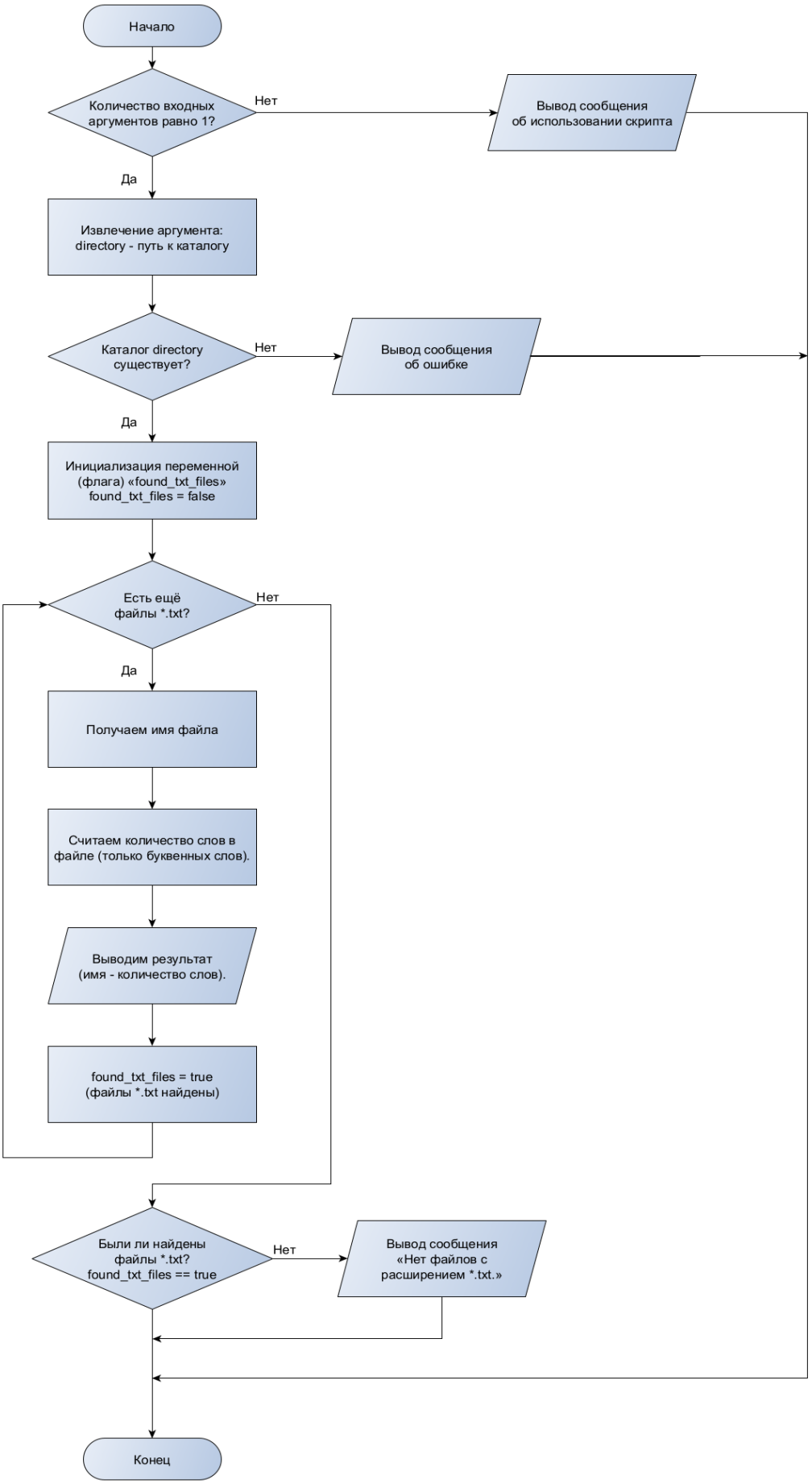

Рисунок 4 – Схема алгоритма программы

В разработанной программе используются следующие библиотеки:

- **stdio.h:** библиотека для работы с файлами, включая функции для открытия и закрытия файлов (fopen, fclose) и ввода-вывода (fprintf, fscanf, fgetc, printf).
- **stdlib.h:** стандартная библиотека языка C, которая предоставляет функции управления памятью (malloc, free) и другие вспомогательные функции (atoi, exit).
- **string.h:** содержит функции для работы со строками, такие как strlen (определение длины строки) и strcmp (сравнение строк).
- **dirent.h:** эта библиотека позволяет работать с директориями, получать список файлов и их атрибуты.
- **ctype.h:** эта библиотека предоставляет функции для работы с символами, такие как isalnum, которая используется для проверки, является ли символ буквой или цифрой.
- **limits.h:** содержит различные системные ограничения, такие как PATH_MAX, которая представляет максимальную длину пути к файлу.

Таблица 1 – Перечень разработанных методов в программе

№	Название	Назначение
1.	is_text_file	Этот метод принимает имя файла и проверяет, является ли он текстовым файлом, проверяя расширение файла. Если имя файла заканчивается на .txt, то он считается текстовым файлом. Также, этот метод имеет проверку на длину имени файла, чтобы исключить слишком короткие имена.
2.	count_words_in_file	Этот метод открывает файл с заданным именем и подсчитывает количество слов в файле. Он считает слово как последовательность букв и цифр, разделенных другими символами. Метод читает файл посимвольно и анализирует каждый символ, чтобы определить границы слов. Он возвращает общее количество слов в файле.
3.	main	Это главная функция программы. Она обрабатывает входные аргументы, открывает указанную директорию, перебирает файлы в этой директории, и для каждого файла, который является текстовым файлом, вызывает count_words_in_file для подсчета слов. После того как все файлы обработаны, программа выводит результат, а также сообщение, если не было найдено файлов с расширением .txt в указанной директории.

Контрольные вопросы:

1) Для считывания аргументов командной строки используются?

Для считывания аргументов командной строки используются аргументы функции main в языке C. В языке C, аргументы командной строки могут быть получены через argc и argv, где argc – это количество аргументов, а argv – массив строк, представляющих аргументы.

2) Как скомпилировать проект?

Для компиляции проекта в языке C используется компилятор GCC с командой вида: gcc -o output_filename source_filename.c, где output_filename - имя файла для создания исполняемого файла, а source_filename.c - исходный файл.

3) Какие функции стандартной библиотеки для работы с файловой системой?

Стандартная библиотека для работы с файловой системой в языке C включает функции, такие как fopen, fclose, fread, fwrite, fseek, ftell и многие другие для открытия, чтения, записи и перемещения по файлам.

4) Как выводить список файлов в два столбца: имя файла, размер файла?

Для вывода списка файлов в два столбца с именем файла и его размером можно использовать команду ls -lh, где -l означает длинный формат, -h означает человеко-читаемый размер.

Приложение А. Bash-скрипт для генерации и заполнении текстовых файлов в указанном каталоге.

```
#!/bin/bash

# Проверяем, что передано два аргумента
if [ "$#" -ne 2 ]; then
    echo "Использование: $0 <путь_к_каталогу> <количество_файлов>"
    exit 1
fi

# Извлекаем аргументы
directory="$1"
file_count="$2"

# Проверяем, что каталог существует
if [ ! -d "$directory" ]; then
    echo "Каталог '$directory' не существует."
    exit 1
fi

# Переходим в указанный каталог
cd "$directory" || exit 1

# Генерируем и записываем файлы
for ((i = 1; i <= file_count; i++)); do
    # Генерируем случайное количество (от 10 до 20) слов
    word_count=$((RANDOM % 11 + 10))

    # Генерируем случайные слова и записываем их в файл
    words=""
    for ((j = 1; j <= word_count; j++)); do
        word=$(cat /dev/urandom | tr -dc 'a-zA-Z0-9' | fold -w 5 | head -n 1)
        words="$words $word"
    done

    # Создаем файл и записываем в него сгенерированные слова
    filename="file_$i.txt"
    echo "$words" > "$filename"
    echo "Создан файл $filename с $word_count словами."
done

echo "Готово! Всего создано $file_count файлов в каталоге '$directory'."
```

Приложение Б. Листинг разработанной программы на Си

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <dirent.h>
#include <ctype.h>
#include <limits.h>

#define MAX_WORD_LENGTH 100

int is_text_file(const char *filename) {
    size_t len = strlen(filename);
    if (len < 4) return 0; // Файлы с короткими именами точно не *.txt
    return (strcmp(filename + len - 4, ".txt") == 0);
}

int count_words_in_file(const char *filename) {
    FILE *file = fopen(filename, "r");
    if (file == NULL) {
        fprintf(stderr, "Не удалось открыть файл: %s\n", filename);
        return -1;
    }

    int word_count = 0;
    int in_word = 0;
    char word[MAX_WORD_LENGTH];
    int word_length = 0;

    int c;
    while ((c = fgetc(file)) != EOF) {
        if (isalnum(c)) {
            if (word_length < MAX_WORD_LENGTH - 1) {
                word[word_length++] = (char) c;
            }
            in_word = 1;
        } else {
            if (in_word) {
                word[word_length] = '\0';
                word_count++;
                in_word = 0;
                word_length = 0;
            }
        }
    }
```

```
    if (in_word) {
        word[word_length] = '\0';
        word_count++;
    }

    fclose(file);
    return word_count;
}

int main(int argc, char *argv[]) {
    if (argc != 2) {
        fprintf(stderr, "Формат вызова программы: %s <путь>\n", argv[0]);
        return 1;
    }

    const char *directory_path = argv[1];
    DIR *dir = opendir(directory_path);
    if (dir == NULL) {
        fprintf(stderr, "Не удалось открыть каталог: %s\n", directory_path);
        return 1;
    }

    struct dirent *entry;
    int found_text_files = 0; // Добавляем переменную для отслеживания .txt файлов
    while ((entry = readdir(dir)) != NULL) {
        if (entry->d_type == DT_REG && is_text_file(entry->d_name)) {
            char filepath[PATH_MAX];
            snprintf(filepath, PATH_MAX, "%s/%s", directory_path, entry->d_name);

            int word_count = count_words_in_file(filepath);
            if (word_count >= 0) {
                printf("%s - %d\n", entry->d_name, word_count);
                found_text_files = 1; // Устанавливаем, что нашли .txt файл
            }
        }
    }

    closedir(dir);

    if (!found_text_files) {
        printf("Нет *.txt файлов в указанной директории.\n");
    }

    return 0;
}
```