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.

1. Разработайте алгоритм программы в соответствии с вариантом задания.
2. Разбейте алгоритм на несколько логических блоков.
3. Создайте исходный текст программы, реализующей разработанный алгоритм.
4. Скомпилируйте проект командой gcc program.c – program, где program — название программы.
5. Устраните синтаксические ошибки, если такие имеются.
6. Разработайте набор тестов с учетом особенностей реализации алгоритма.
7. Проверьте правильность функционирования программы с использованием набора тестов.
8. 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.

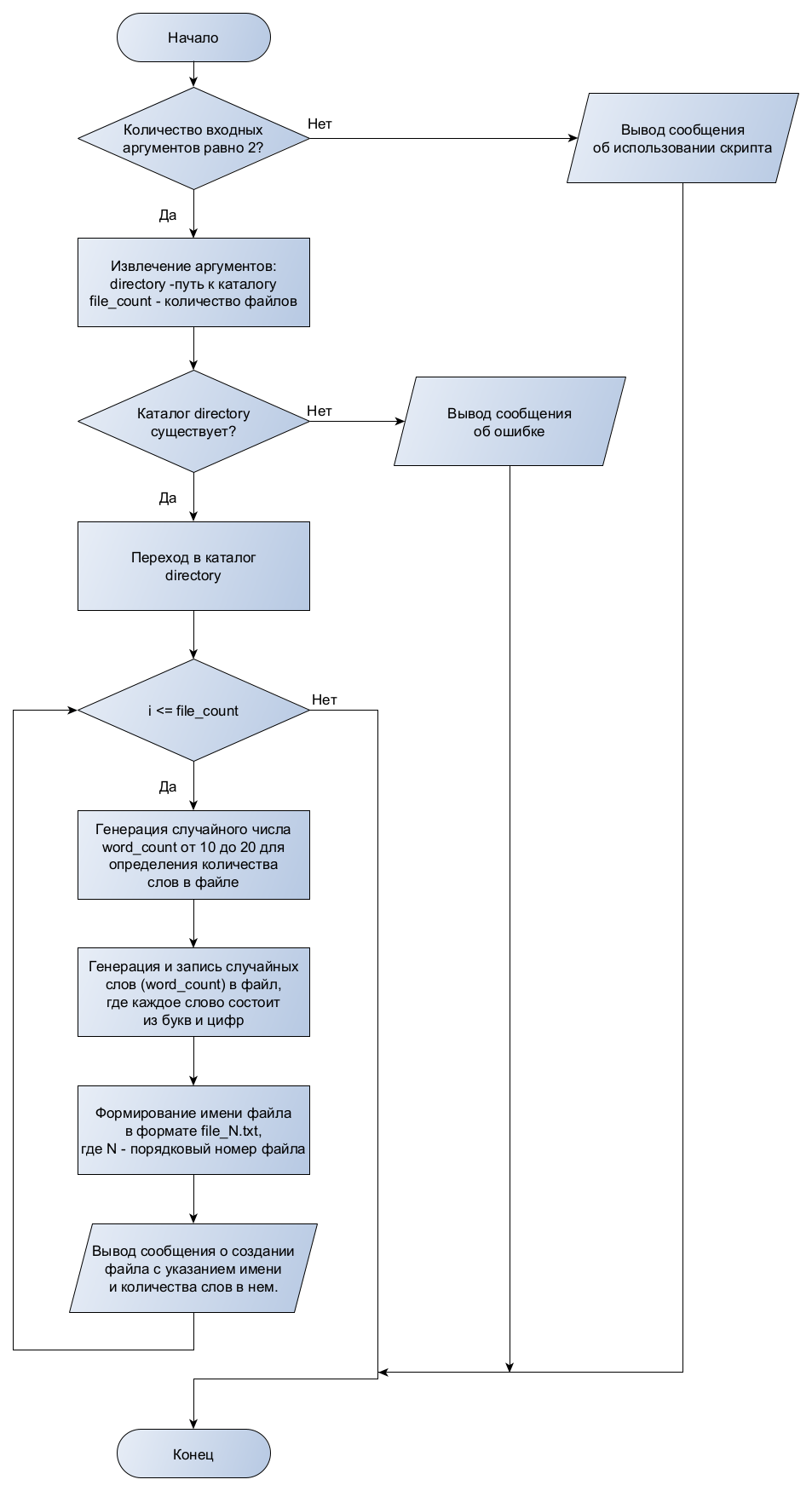


Рисунок 1 – схема алгоритма генерации текстовых файлов

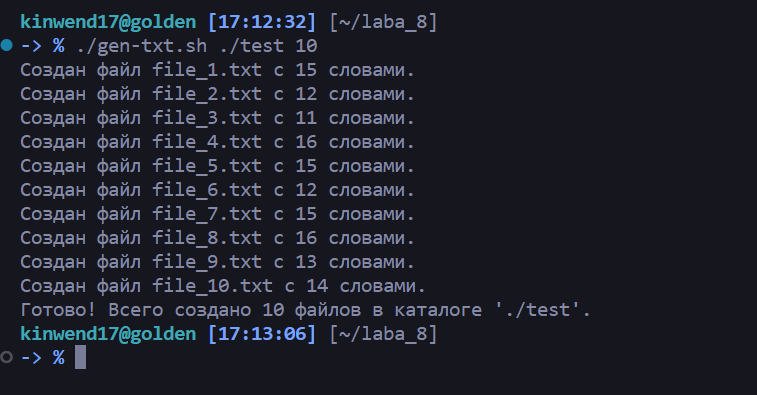


Рисунок 2 – Пример вывода bash-скрипта генерации файлов

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>.

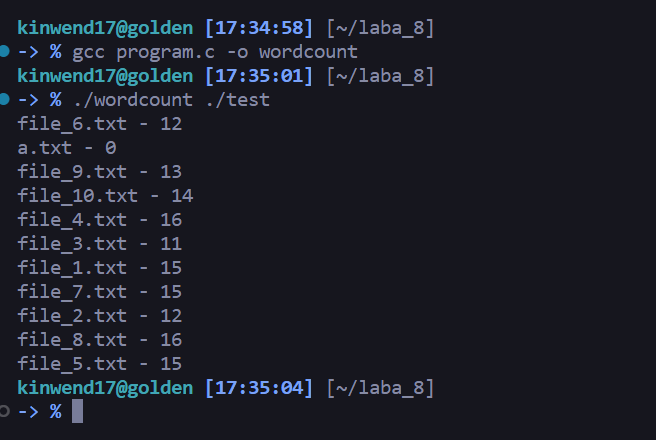


Рисунок 3 – Вывод разработанной программы

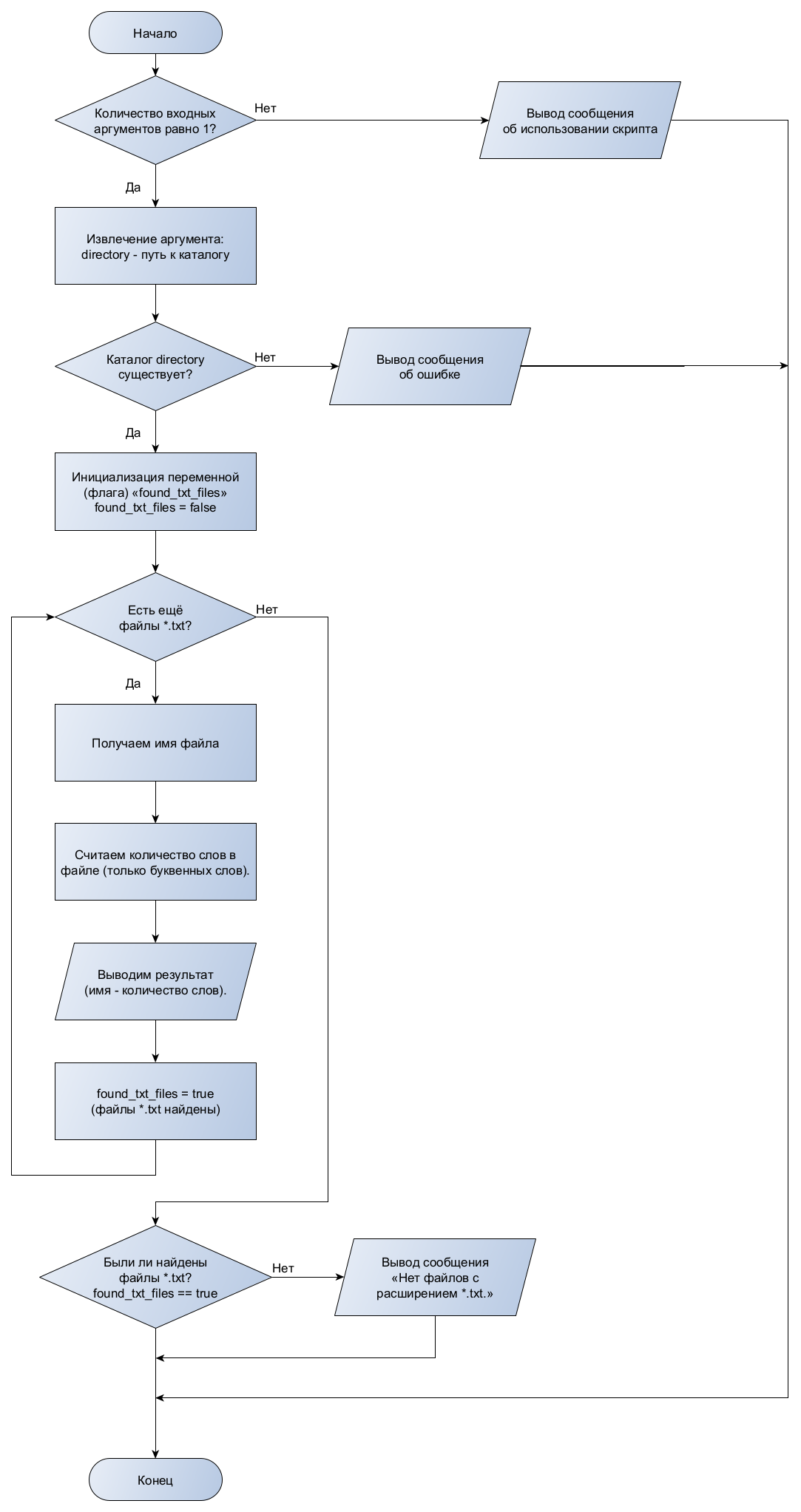


Рисунок 4 – Схема алгоритма программы

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 |
|  | 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. |
|  | 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. |
|  | 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.

1. 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.

1. 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.

1. 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;

}