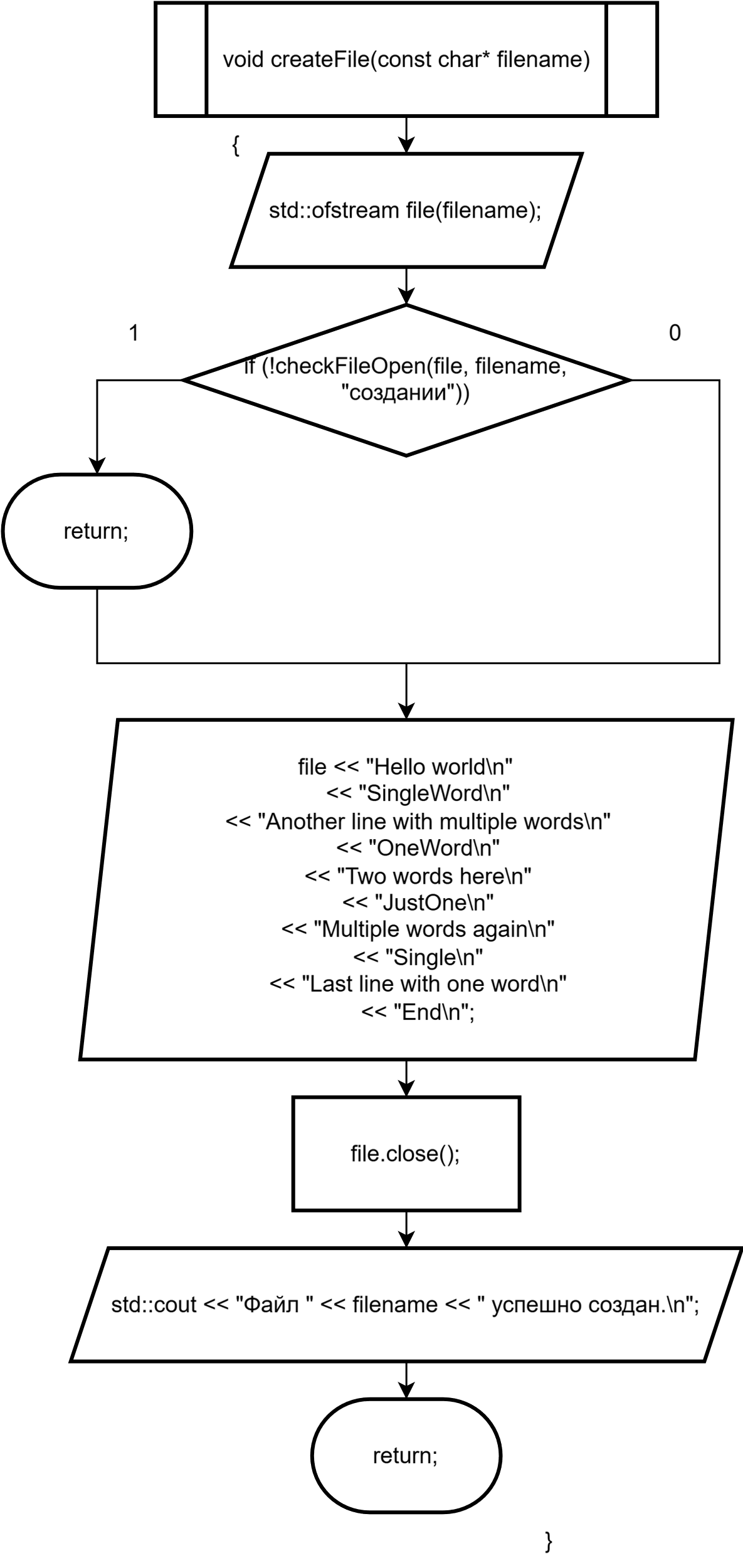
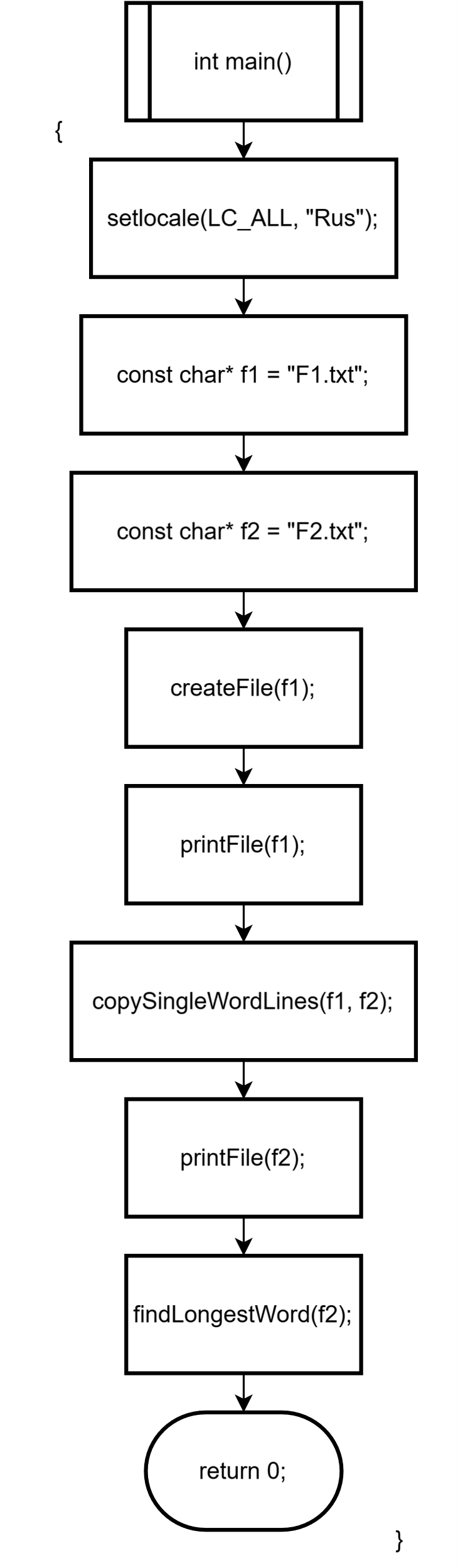
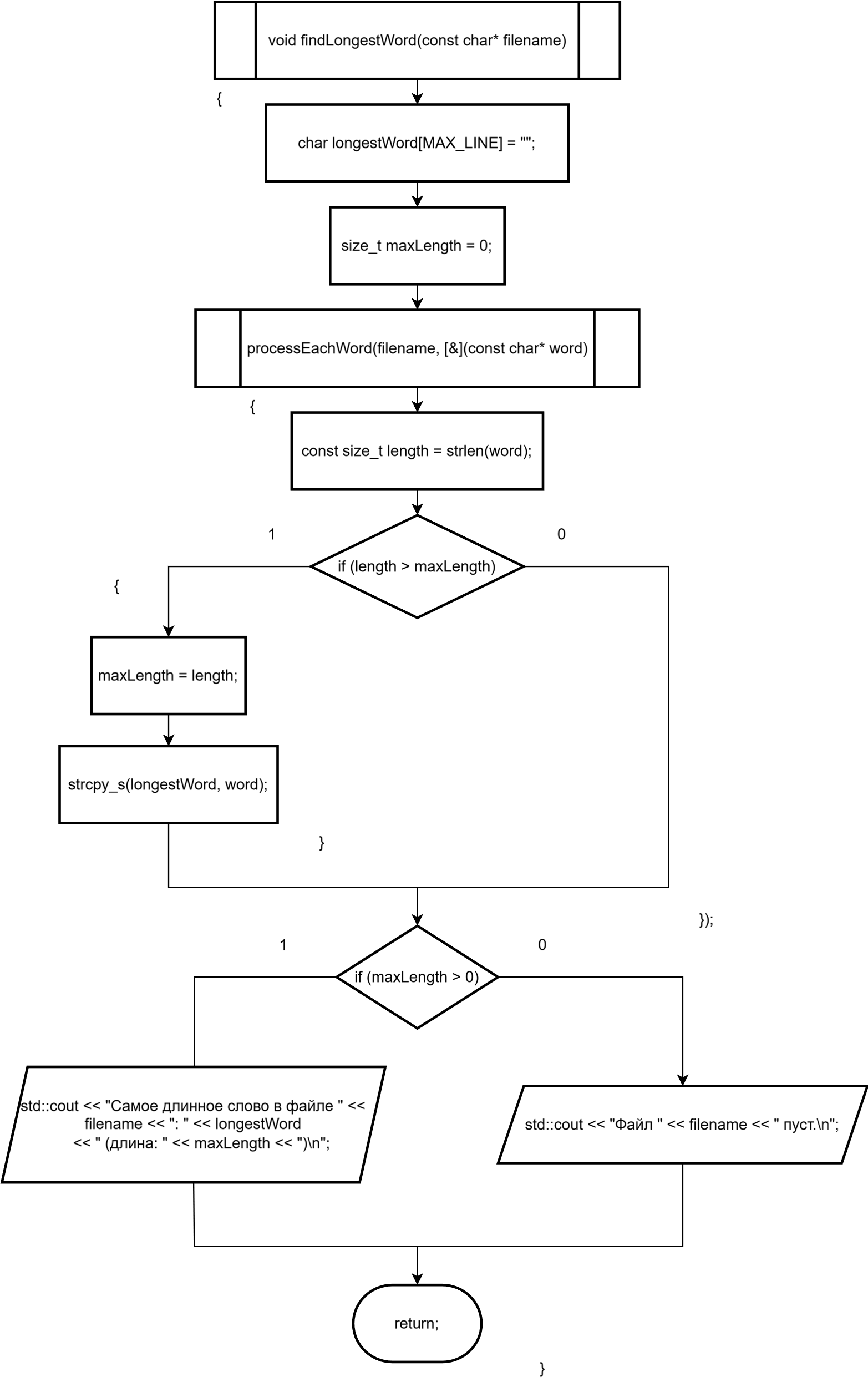
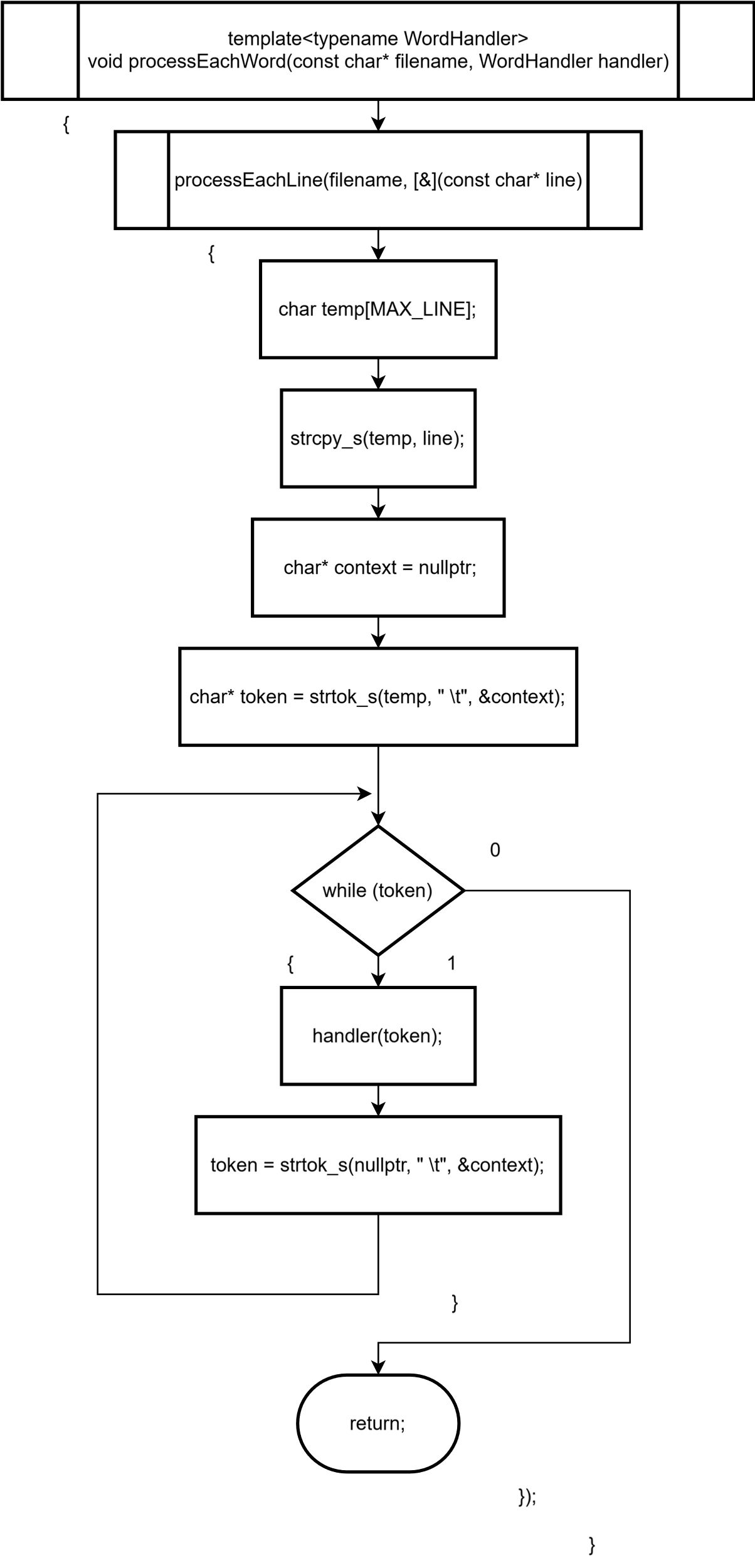
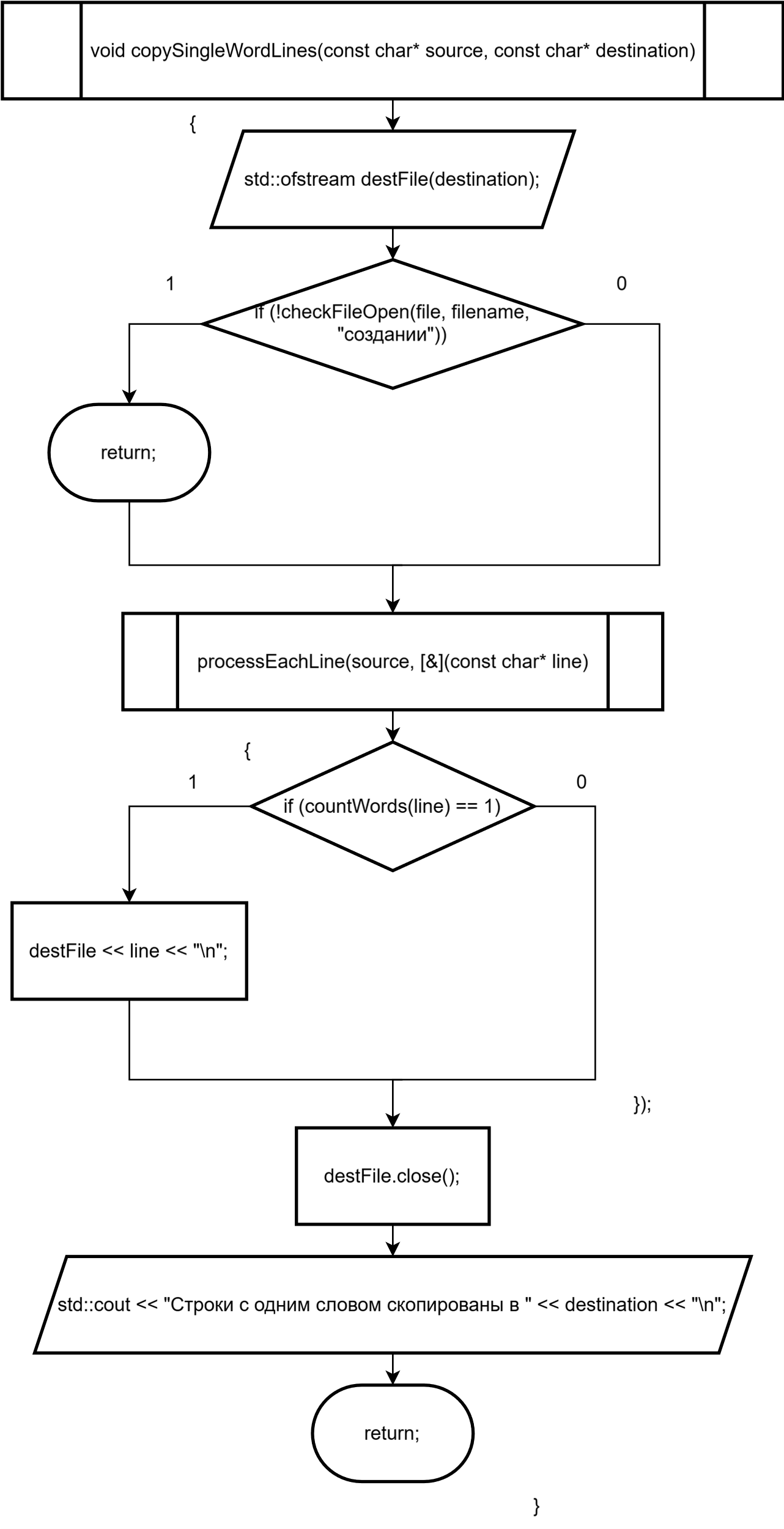
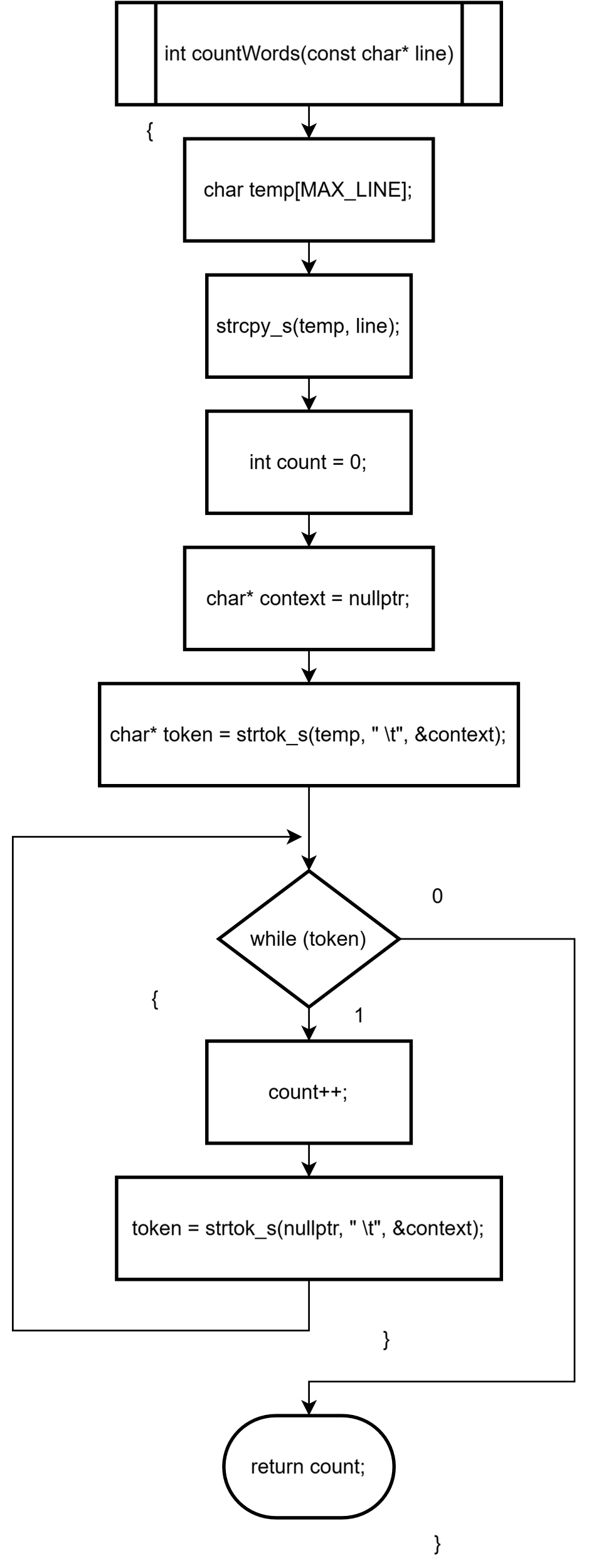
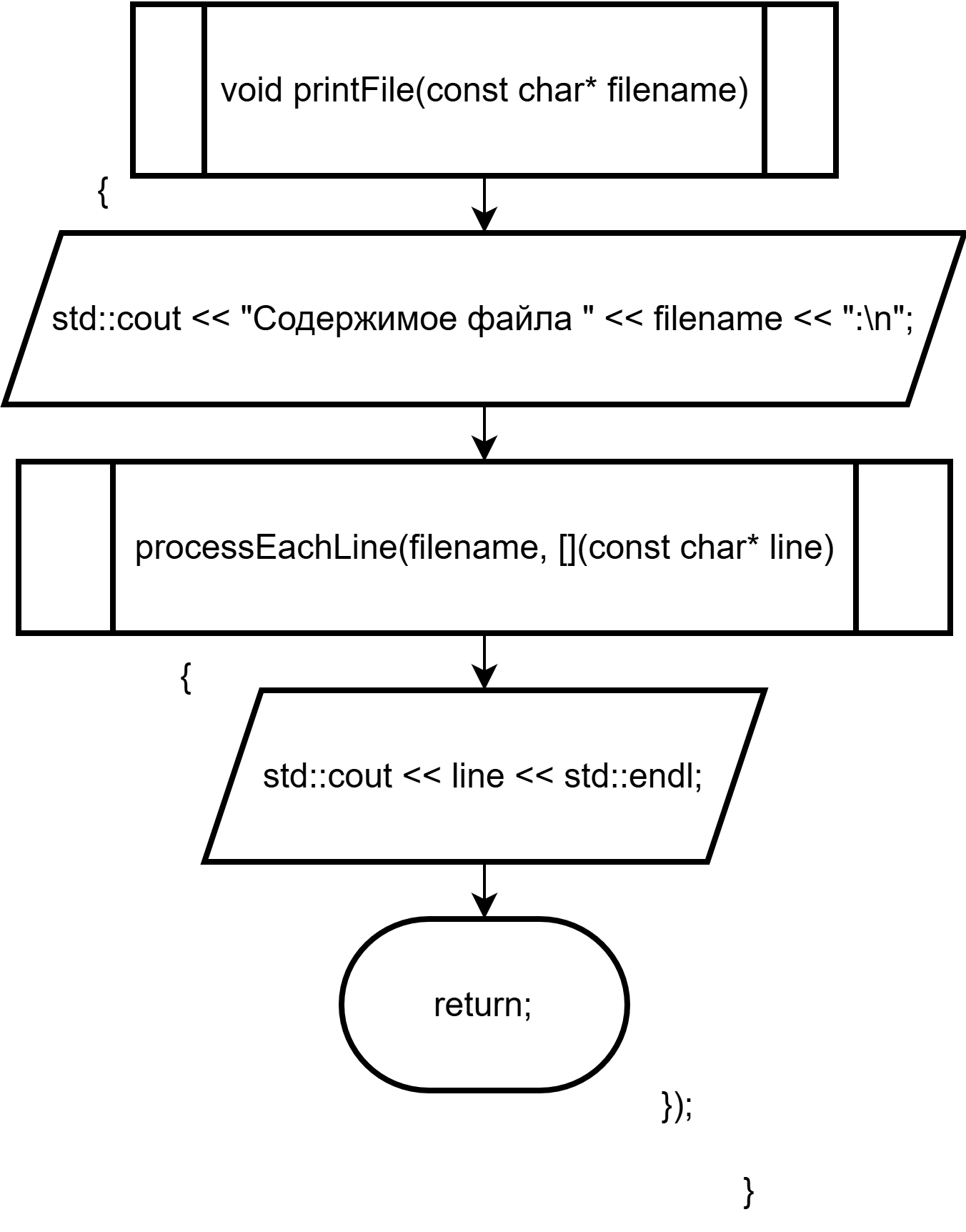
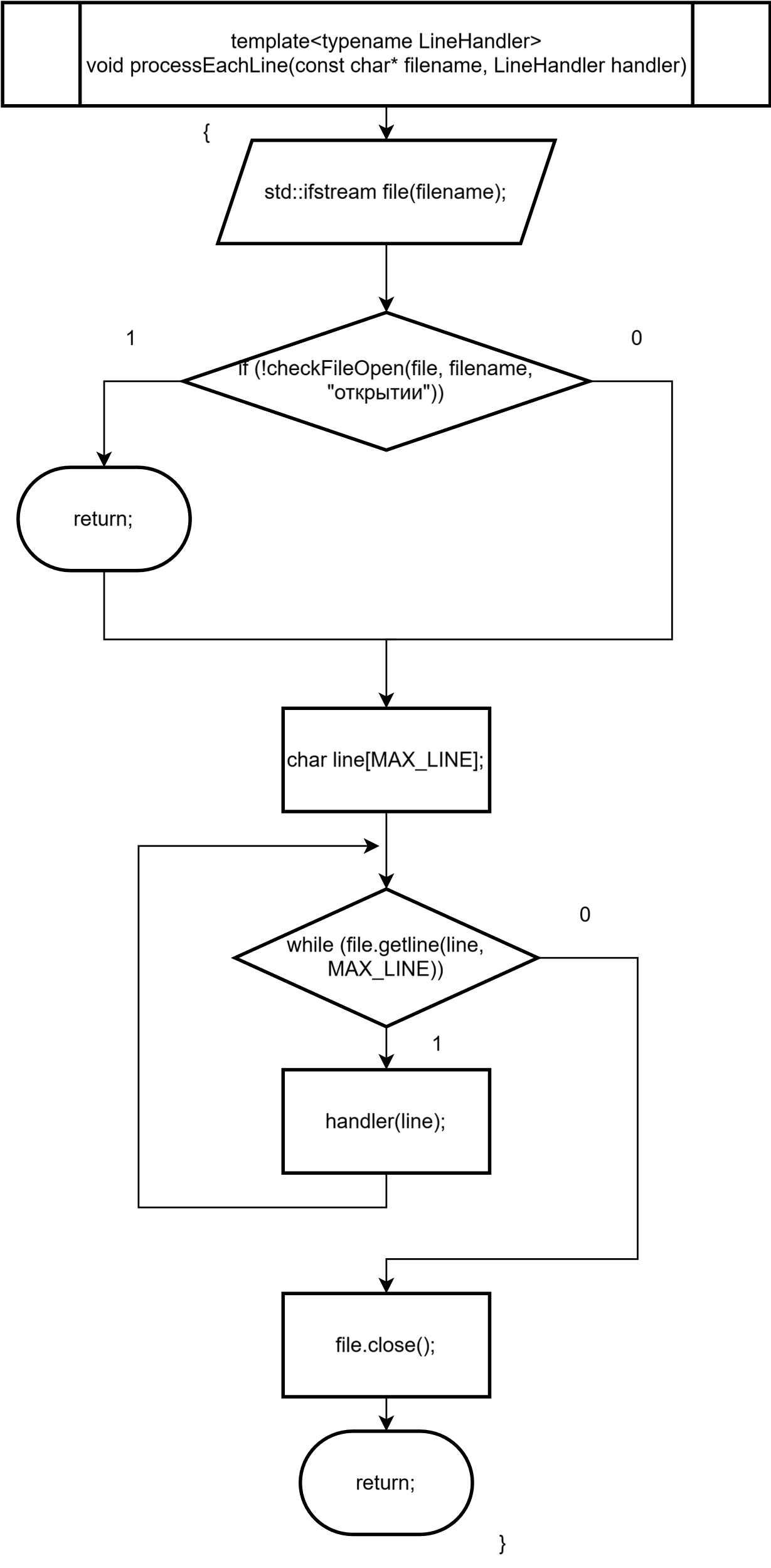
|  |  |
| --- | --- |
| Пермский национальный исследовательский политехнический университет  Кафедра информационные технологии и автоматизированные системы | |
|  |  |
|  |  |
| Лабораторная работа №9  «Строковый ввод-вывод»  Вариант №9 | |
|  |  |
|  | Выполнил:  Студент группы ИВТ-24-2б  А.С. Ваулин  Проверил:  Доцент кафедры ИТАС  О.А. Полякова |
|  |  |
| Пермь 2025 | |

# Постановка задачи

1. Создать текстовый файл F1 не менее, чем из 10 строк и записать в него информацию
2. Скопировать из файла F1 в файл F2 все строки, которые содержат только одно слово.
3. Найти самое длинное слово в файле F2.

# Блок-схема





# Код программы

#include <iostream>

#include <fstream>

const int MAX\_LINE = 255;

// Общая функция проверки открытия файла

template<typename TStream>

bool checkFileOpen(TStream& stream, const char\* filename, const char\* mode) {

if (!stream.is\_open()) {

std::cerr << "Ошибка при " << mode << " файла " << filename << std::endl;

return false;

}

return true;

}

// Функция для формирования файла

void createFile(const char\* filename) {

std::ofstream file(filename);

if (!checkFileOpen(file, filename, "создании")) return;

file << "Hello world\n"

<< "SingleWord\n"

<< "Another line with multiple words\n"

<< "OneWord\n"

<< "Two words here\n"

<< "JustOne\n"

<< "Multiple words again\n"

<< "Single\n"

<< "Last line with one word\n"

<< "End\n";

file.close();

std::cout << "Файл " << filename << " успешно создан.\n";

}

// Общая функция обработки строк

template<typename LineHandler>

void processEachLine(const char\* filename, LineHandler handler) {

std::ifstream file(filename);

if (!checkFileOpen(file, filename, "открытии")) return;

char line[MAX\_LINE];

while (file.getline(line, MAX\_LINE)) {

handler(line);

}

file.close();

}

// Функция для печати файла

void printFile(const char\* filename) {

std::cout << "Содержимое файла " << filename << ":\n";

processEachLine(filename, [](const char\* line) {

std::cout << line << std::endl;

});

}

// Функция для подсчета слов в строке

int countWords(const char\* line) {

char temp[MAX\_LINE];

strcpy\_s(temp, line);

int count = 0;

char\* context = nullptr;

char\* token = strtok\_s(temp, " \t", &context);

while (token) {

count++;

token = strtok\_s(nullptr, " \t", &context);

}

return count;

}

// Функция для копирования строк с одним словом

void copySingleWordLines(const char\* source, const char\* destination) {

std::ofstream destFile(destination);

if (!checkFileOpen(destFile, destination, "создании")) return;

processEachLine(source, [&](const char\* line) {

if (countWords(line) == 1) {

destFile << line << "\n";

}

});

destFile.close();

std::cout << "Строки с одним словом скопированы в " << destination << "\n";

}

// Общая функция обработки слов

template<typename WordHandler>

void processEachWord(const char\* filename, WordHandler handler) {

processEachLine(filename, [&](const char\* line) {

char temp[MAX\_LINE];

strcpy\_s(temp, line);

char\* context = nullptr;

char\* token = strtok\_s(temp, " \t", &context);

while (token) {

handler(token);

token = strtok\_s(nullptr, " \t", &context);

}

});

}

// Функция для нахождения самого длинного слова

void findLongestWord(const char\* filename) {

char longestWord[MAX\_LINE] = "";

size\_t maxLength = 0;

processEachWord(filename, [&](const char\* word) {

const size\_t length = strlen(word);

if (length > maxLength) {

maxLength = length;

strcpy\_s(longestWord, word);

}

});

if (maxLength > 0) {

std::cout << "Самое длинное слово в файле " << filename << ": " << longestWord

<< " (длина: " << maxLength << ")\n";

}

else {

std::cout << "Файл " << filename << " пуст.\n";

}

}

int main() {

setlocale(LC\_ALL, "Rus");

const char\* f1 = "F1.txt";

const char\* f2 = "F2.txt";

createFile(f1);

printFile(f1);

copySingleWordLines(f1, f2);

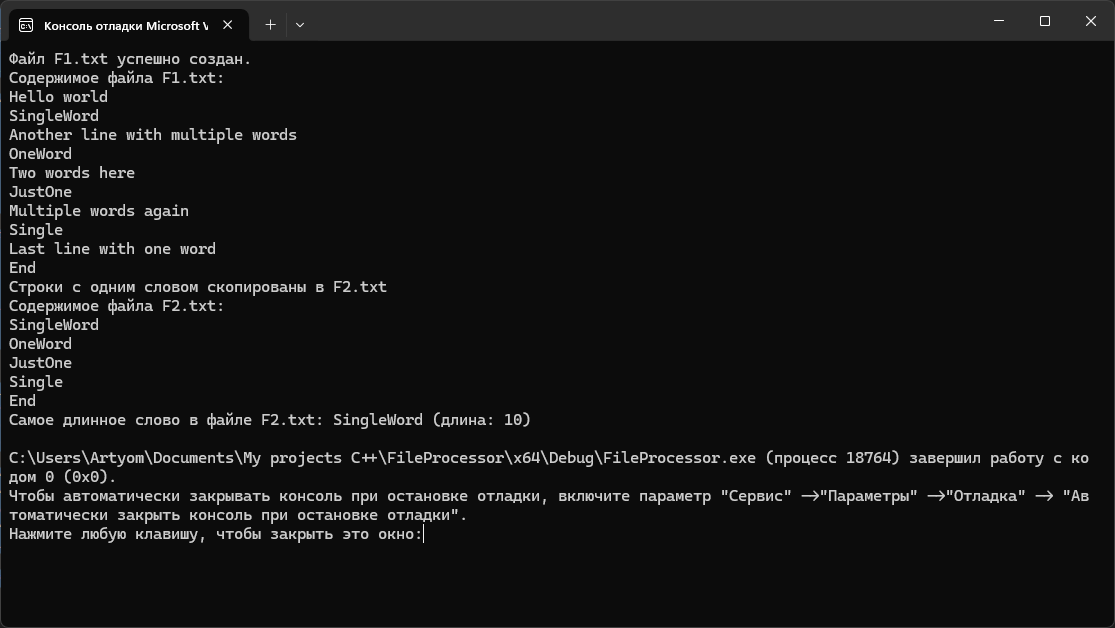
printFile(f2);

findLongestWord(f2);

return 0;

}

# Результат работы



Git: <https://github.com/Artivaa/FileProcessor>