Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования «Тихоокеанский государственный университет»

Высшая школа кибернетики и цифровых технологий

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №3

ПО КУРСУ «ОСНОВЫ ПРОГРАММНОЙ ИНЖЕНЕРИИ»

«ДОКУМЕНТИРОВАНИЕ КОДА»

Выполнил: студент 2 курса группы ПО(б)-31

Гнездилов Кирилл Романович

Проверила: ассистент ВШ КЦТ

Балабасова Алиса Дмитриевна

Хабаровск 2025 г.

**Цель работы:** Освоить работу с инструментами автоматической генерации документации.

**Задание:**

1. Добавить комментарии в код по стандарту Doxygen;
2. Сгенерировать HTML-документацию;
3. Настроить автоматическое обновление документации при коммите.

**Ход работы:**

В качестве тестового проекта был взят программный комплекс из лабораторной работы 1 по "Основам программной инженерии" прошлого семестра.

Для начала я установил Doxygen на компьютер. Далее в каждом файле программы добавил комментарии по стандарту Doxygen (листинги 1, 2, 3).

Листинг 1 (fraction.h):

/// \file fraction.h

/// \brief Заголовочный файл для работы с дробями

#pragma once

#include <string>

/// \struct Fraction

/// \brief Представляет дробь в виде числителя/знаменателя

struct Fraction {

int numerator; ///< Числитель дроби

int denominator; ///< Знаменатель дроби (не может быть 0)

};

/// \brief Создает корректную дробь

/// \param numerator Числитель

/// \param denominator Знаменатель

/// \return Структура Fraction

/// \exception std::invalid\_argument Если знаменатель = 0

Fraction createFraction(int numerator, int denominator);

/// \brief Сложение двух дробей

/// \param a Первая дробь

/// \param b Вторая дробь

/// \return Результат сложения (сокращенная дробь)

Fraction addFractions(Fraction a, Fraction b);

/// \brief Вычитание двух дробей

/// \param a Уменьшаемое (дробь)

/// \param b Вычитаемое (дробь)

/// \return Результат вычитания (сокращенная дробь)

Fraction subtractFractions(Fraction a, Fraction b);

/// \brief Умножение двух дробей

/// \param a Первая дробь

/// \param b Вторая дробь

/// \return Результат умножения (сокращенная дробь)

Fraction multiplyFractions(Fraction a, Fraction b);

/// \brief Деление двух дробей

/// \param a Делимое (дробь)

/// \param b Делитель (дробь)

/// \return Результат деления (сокращенная дробь)

/// \exception std::invalid\_argument Если числитель делителя = 0

Fraction divideFractions(Fraction a, Fraction b);

/// \brief Сокращение дроби

/// \param f Исходная дробь

/// \return Сокращенная дробь

Fraction reduceFraction(Fraction f);

/// \brief Преобразование дроби в строку

/// \param f Дробь для преобразования

/// \return Строковое представление дроби (вида "a/b" или "a")

std::string fractionToString(Fraction f);

Листинг 2 (fraction.cpp):

/// \file fraction.cpp

/// \brief Реализация операций с дробями

#include "fraction.h"

#include <stdexcept>

#include <algorithm>

#include <sstream>

using namespace std;

/// \brief Вычисление наибольшего общего делителя (НОД) по алгоритму Евклида

/// \param a Первое число

/// \param b Второе число

/// \return Наибольший общий делитель

int gcd(int a, int b) {

return b == 0 ? a : gcd(b, a % b);

}

/// \brief Создает дробь с проверкой знаменателя и нормализацией знака

/// \param numerator Числитель

/// \param denominator Знаменатель

/// \return Структура Fraction

/// \exception std::invalid\_argument Если знаменатель = 0

Fraction createFraction(int numerator, int denominator) {

if (denominator == 0) {

throw invalid\_argument("Знаменатель не может быть нулем");

}

if (denominator < 0) {

numerator = -numerator;

denominator = -denominator;

}

return {numerator, denominator};

}

/// \brief Сокращение дроби с использованием НОД

/// \param f Исходная дробь

/// \return Сокращенная дробь

Fraction reduceFraction(Fraction f) {

int common\_divisor = gcd(abs(f.numerator), abs(f.denominator));

return {

f.numerator / common\_divisor,

f.denominator / common\_divisor

};

}

/// \brief Сложение дробей: (a/b) + (c/d) = (ad + bc)/bd

/// \param a Первая дробь

/// \param b Вторая дробь

/// \return Результат сложения (сокращенная дробь)

Fraction addFractions(Fraction a, Fraction b) {

int num = a.numerator \* b.denominator + b.numerator \* a.denominator;

int denom = a.denominator \* b.denominator;

return reduceFraction({num, denom});

}

/// \brief Вычитание дробей: (a/b) - (c/d) = (ad - bc)/bd

/// \param a Уменьшаемое (дробь)

/// \param b Вычитаемое (дробь)

/// \return Результат вычитания (сокращенная дробь)

Fraction subtractFractions(Fraction a, Fraction b) {

int num = a.numerator \* b.denominator - b.numerator \* a.denominator;

int denom = a.denominator \* b.denominator;

return reduceFraction({num, denom});

}

/// \brief Умножение дробей: (a/b) \* (c/d) = ac/bd

/// \param a Первая дробь

/// \param b Вторая дробь

/// \return Результат умножения (сокращенная дробь)

Fraction multiplyFractions(Fraction a, Fraction b) {

int num = a.numerator \* b.numerator;

int denom = a.denominator \* b.denominator;

return reduceFraction({num, denom});

}

/// \brief Деление дробей: (a/b) / (c/d) = ad/bc

/// \param a Делимое (дробь)

/// \param b Делитель (дробь)

/// \return Результат деления (сокращенная дробь)

/// \exception std::invalid\_argument Если числитель делителя = 0

Fraction divideFractions(Fraction a, Fraction b) {

if (b.numerator == 0) {

throw invalid\_argument("Деление на ноль");

}

int num = a.numerator \* b.denominator;

int denom = a.denominator \* b.numerator;

return reduceFraction({num, denom});

}

/// \brief Преобразование дроби в строковое представление

/// \param f Дробь для преобразования

/// \return Строка вида "числитель/знаменатель" или "целое число" (если знаменатель = 1)

std::string fractionToString(Fraction f) {

if (f.denominator == 1) {

return to\_string(f.numerator);

}

return to\_string(f.numerator) + "/" + to\_string(f.denominator);

}

Листинг 3 (main.cpp):

#include <iostream>

#include <fstream>

#include <string>

#include <vector>

#include <locale>

#include "fraction.h"

using namespace std;

// Функция для отображения справочной информации

void showHelp() {

cout << "Доступные команды:" << endl;

cout << " --help показать это меню" << endl;

cout << " --op <операция> выполнить операцию:" << endl;

cout << " create создать дробь" << endl;

cout << " add сложить две дроби" << endl;

cout << " sub вычесть две дроби" << endl;

cout << " mul умножить две дроби" << endl;

cout << " div разделить две дроби" << endl;

cout << " reduce сократить дробь" << endl;

cout << " print вывести дробь" << endl;

}

// Функция чтения входных данных из файла input.txt

vector<string> readInputData() {

ifstream inputFile("input.txt");

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

throw runtime\_error("Не удалось открыть файл input.txt");

}

vector<string> tokens;

string token;

while (inputFile >> token) {

tokens.push\_back(token);

}

inputFile.close();

return tokens;

}

// Функция записи результата в файл output.txt

void writeOutputData(const string& result) {

ofstream outputFile("output.txt");

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

throw runtime\_error("Не удалось открыть файл output.txt");

}

outputFile << result;

outputFile.close();

}

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

setlocale(LC\_ALL, "Russian");

locale::global(locale(""));

// Если нет аргументов командной строки, показать помощь и выйти

if (argc == 1) {

showHelp();

return 0;

}

string operation;

bool hasOperation = false;

// Обработка аргументов командной строки

for (int i = 1; i < argc; ++i) {

string arg = argv[i];

if (arg == "--help") {

showHelp();

return 0;

} else if (arg == "--op") {

if (i + 1 < argc) {

operation = argv[++i];

hasOperation = true;

} else {

cerr << "Ошибка: после --op требуется указать операцию" << endl;

showHelp();

return 1;

}

}

}

// Если не указана операция, показать помощь и выйти

if (!hasOperation) {

showHelp();

return 0;

}

try {

vector<string> inputData = readInputData();

string result;

if (operation == "create") {

if (inputData.size() < 2) throw runtime\_error("Нужно 2 числа");

Fraction f = createFraction(stoi(inputData[0]), stoi(inputData[1]));

result = fractionToString(f);

}

else if (operation == "add") {

if (inputData.size() < 4) throw runtime\_error("Нужно 4 числа");

Fraction a = createFraction(stoi(inputData[0]), stoi(inputData[1]));

Fraction b = createFraction(stoi(inputData[2]), stoi(inputData[3]));

result = fractionToString(addFractions(a, b));

}

else if (operation == "sub") {

if (inputData.size() < 4) throw runtime\_error("Нужно 4 числа");

Fraction a = createFraction(stoi(inputData[0]), stoi(inputData[1]));

Fraction b = createFraction(stoi(inputData[2]), stoi(inputData[3]));

result = fractionToString(subtractFractions(a, b));

}

else if (operation == "mul") {

if (inputData.size() < 4) throw runtime\_error("Нужно 4 числа");

Fraction a = createFraction(stoi(inputData[0]), stoi(inputData[1]));

Fraction b = createFraction(stoi(inputData[2]), stoi(inputData[3]));

result = fractionToString(multiplyFractions(a, b));

}

else if (operation == "div") {

if (inputData.size() < 4) throw runtime\_error("Нужно 4 числа");

Fraction a = createFraction(stoi(inputData[0]), stoi(inputData[1]));

Fraction b = createFraction(stoi(inputData[2]), stoi(inputData[3]));

result = fractionToString(divideFractions(a, b));

}

else if (operation == "reduce") {

if (inputData.size() < 2) throw runtime\_error("Нужно 2 числа");

Fraction f = createFraction(stoi(inputData[0]), stoi(inputData[1]));

result = fractionToString(reduceFraction(f));

}

else if (operation == "print") {

if (inputData.size() < 2) throw runtime\_error("Нужно 2 числа");

Fraction f = createFraction(stoi(inputData[0]), stoi(inputData[1]));

cout << "Результат: " << fractionToString(f) << endl;

return 0;

}

else {

throw runtime\_error("Неизвестная операция: " + operation);

}

writeOutputData(result);

cout << "Результат записан в output.txt" << endl;

} catch (const exception& e) {

cerr << "Ошибка: " << e.what() << endl;

return 1;

}

return 0;

}

Далее я создал конфигурационный файл Doxyfile, добавив основные параметры (рисунок 1).

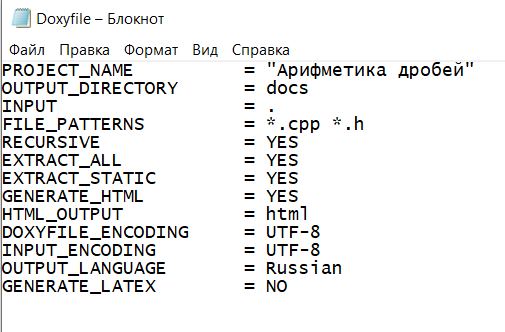


Рисунок 1 - Содержимое Doxyfile

PROJECT\_NAME - Название проекта, которое будет отображаться в документации.

OUTPUT\_DIRECTORY - Папка, в которую будет сохраняться сгенерированная документация (в данном случае — docs).

FILE\_PATTERNS - шаблоны имен файлов, которые будут обрабатываться.

INPUT Указывает Doxygen анализировать исходные файлы в текущей директории (.).

RECURSIVE - Включён рекурсивный поиск файлов в поддиректориях.

EXTRACT\_ALL = YES - Doxygen будет документировать все элементы, даже если они не имеют специальных комментариев.

EXTRACT\_STATIC = YES - Будут документированы статические члены и функции.

GENERATE\_HTML = YES - Включена генерация HTML-документации.

HTML\_OUTPUT = html - HTML-файлы будут сохранены в поддиректорию html внутри OUTPUT\_DIRECTORY.

DOXYFILE\_ENCODING - Кодировка самого конфигурационного файла.

INPUT\_ENCODING = UTF-8 - Кодировка исходных файлов, которые анализирует Doxygen.

OUTPUT\_LANGUAGE - Язык выходной документации.

GENERATE\_LATEX - генерация документация в формате PDF.

Теперь могу генерировать HTML-документацию. Введу в терминал команду "doxygen Doxyfile", после выполнения которой появляется папка docs, а в ней - множество файлов (рисунок 2).

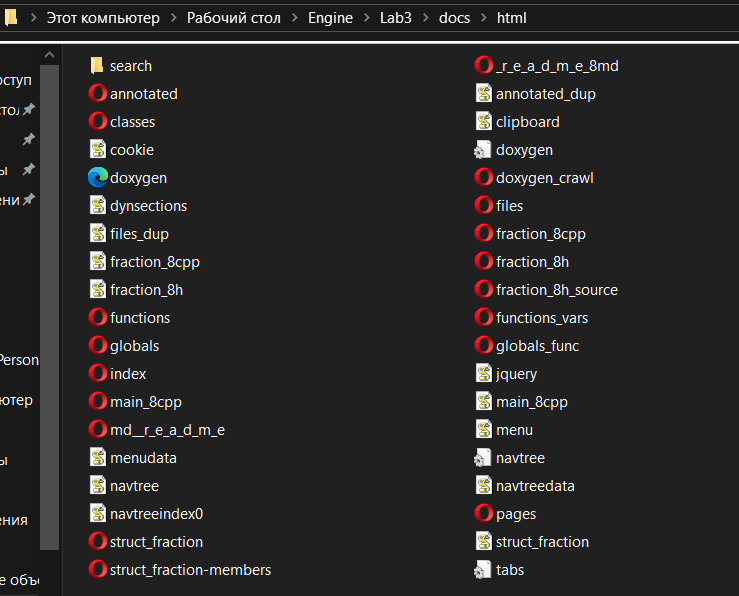


Рисунок 2 - Генерация HTML-документации

Теперь, открыв файл docs/html/index.html открываю документацию в браузере (рисунок 3).

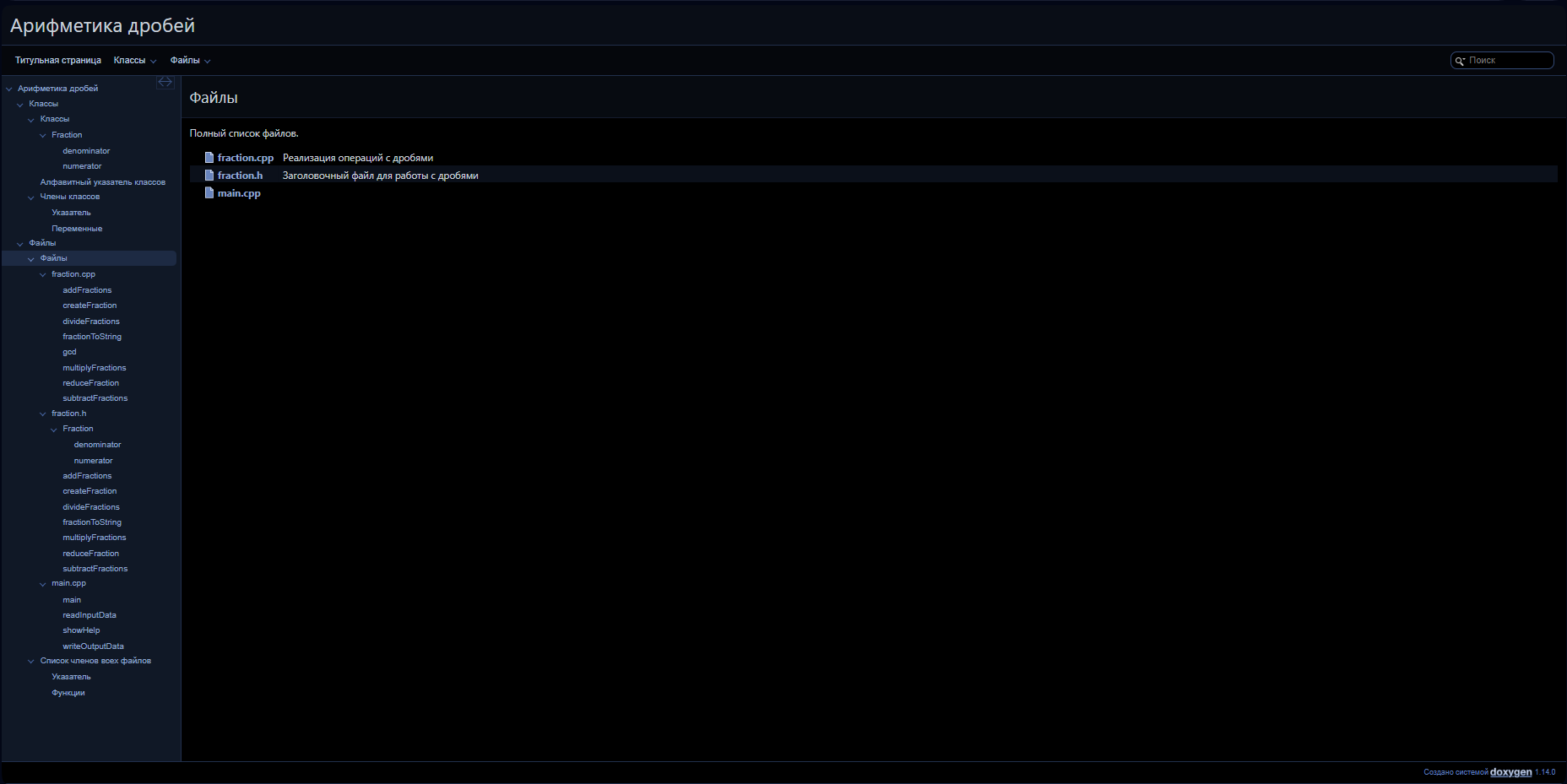


Рисунок 3 - Открытая в браузере HTML-документация

Создам отдельный репозиторий для этого проекта (рисунок 4).

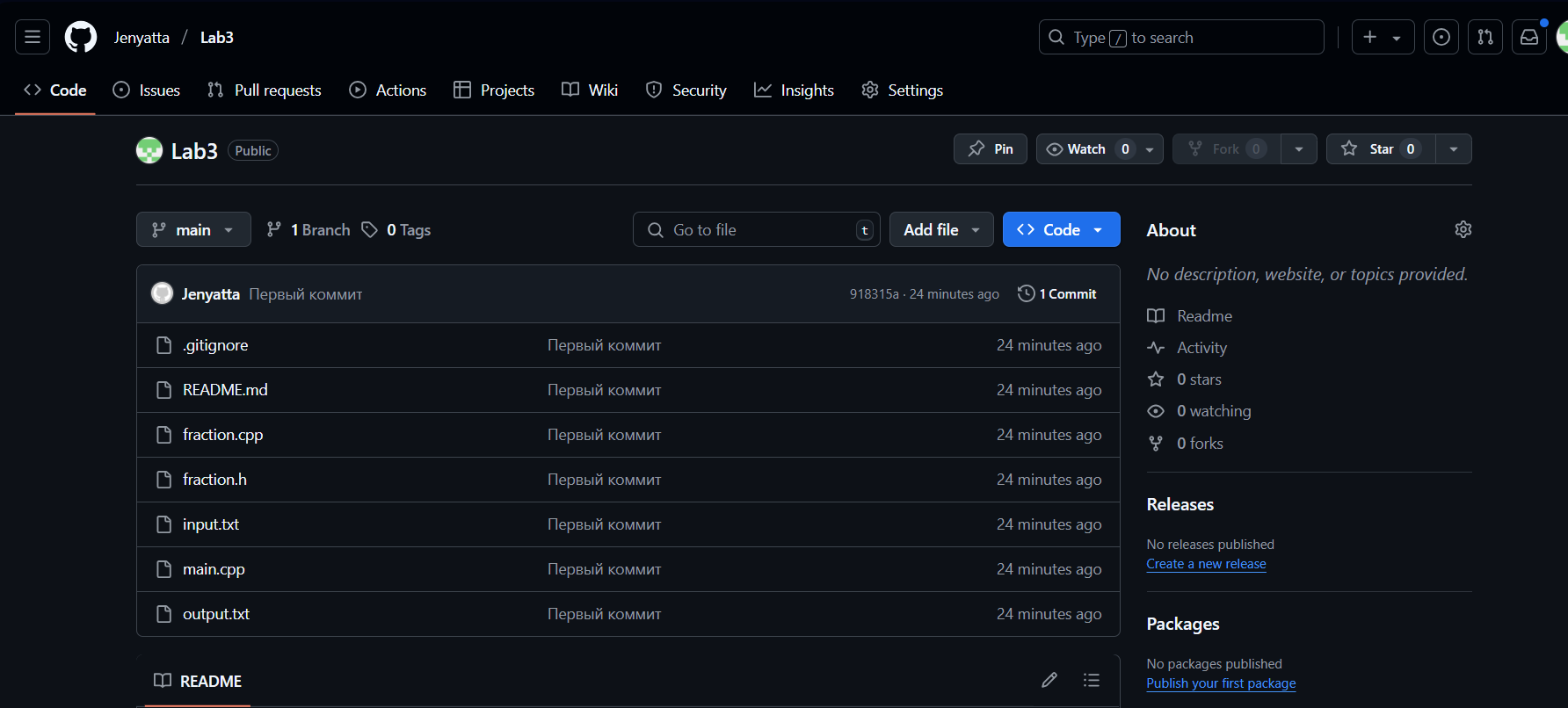


Рисунок 4 - Репозиторий для проекта

Далее я создаю в папке .git/hooks файл post-commit - этот файл будет автоматически выполняться при каждом коммите (рисунок 5).

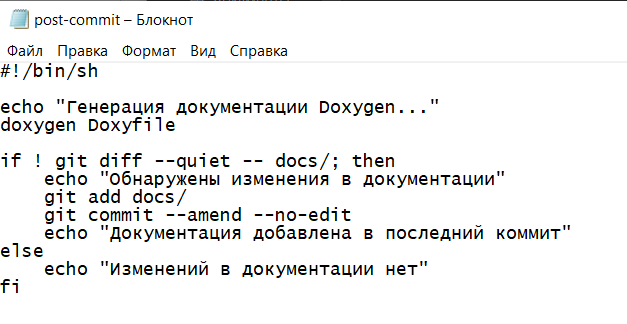


Рисунок 5 - Содержимое файла post-commit

Чтобы все работало, нужно файлу post-commit дать права на выполнение командой chmod +x .git/hooks/post-commit. Так как я использую операционную систему Windows 10, то я использую для данной команды Git Bush.

Теперь при каждом коммите автоматически будет обновляться документация. Для демонстрации я просто попробую убрать \*.cpp из шаблонов имен файлов, которые обрабатываются Doxygen. Теперь сделаю коммит и проверю HTML-документацию, информация о файлах fraction.cpp и main.cpp должна пропасть (рисунок 6).

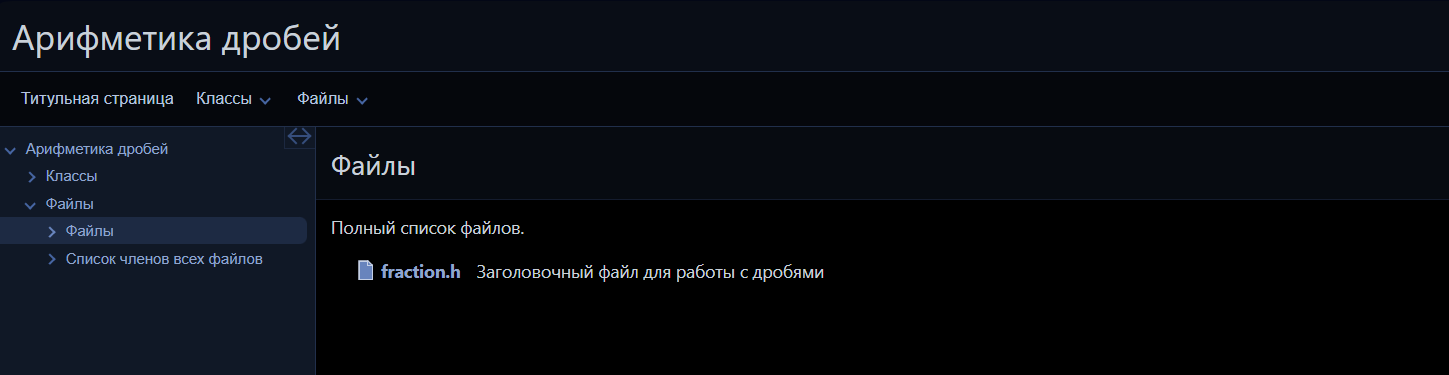


Рисунок 6 - Изменение документации

**Вывод:** В ходе лабораторной работы я изучил основные функции и освоил настройку системы автоматической генерации документации Doxygen.