МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ государственное БЮДЖЕТНОЕ

образовательное учреждение

высшего образования

«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кафедра защиты информации



**ОТЧЁТ**

**по лабораторной работе №1**

**по дисциплине: «***Программирование***»**

Выполнил:Проверил:

Студент гр. «АБс-426», «АВТФ» *Ассистент кафедры ЗИ*

*Петроченко М. Р. Исаев Г. А.*

«\_\_» \_\_\_\_\_\_ 2025 г.«\_\_\_» \_\_\_\_\_\_ 2025 г.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись) (подпись)

Новосибирск 2025

**Цели и задачи работы**: изучение платформы GitHub и практическое

использование Git для хранения, обновления и распространения исходного

кода проекта.

**Задание №1 к работе**: Разработать алгоритм решения задачи по индивидуальному заданию.

Реализовать функции, связанные со строками:

c. Трапеция – периметр, площадь, длина средней линии

**Методика выполнения работы**:

1. Разработать алгоритм решения задачи по индивидуальному заданию.
2. Написать и отладить программу решения задачи.
3. Протестировать работу программы на различных исходных данных.
4. Продемонстрировать работу с Git и GitHub на примере написанной программы.
5. По запросу преподавателя быть готовым модифицировать/добавить функционал программы.
6. Ответить на теоретические вопросы к лабораторной работе на выбор преподавателя.

**Листинг программы**:

<https://github.com/ri2rixxx/laboratornaya_rabota_1>

#include <iostream>

#include <cmath>

#include <iomanip>

using namespace std;

struct Trapezoid {

double a, b, c, d, h;

double middleLine() {

return (a + b) / 2;

}

double area() {

return middleLine() \* h;

}

double perimeter() {

return a + b + c + d;

}

// Проверка существования трапеции

bool isValid() {

// 1. Высота должна быть меньше боковых сторон

if (h >= c || h >= d) {

cerr << "Ошибка: высота должна быть меньше боковых сторон!\n";

return false;

}

// 2. Сумма боковых сторон должна быть больше разности оснований

double minBase = min(a, b);

double maxBase = max(a, b);

double sideSum = c + d;

double baseDiff = maxBase - minBase;

if (sideSum <= baseDiff) {

cerr << "Ошибка: сумма боковых сторон должна быть больше разности оснований!\n";

return false;

}

// 3. Проверка, что высота не слишком маленькая при больших боковых сторонах

// Используем теорему Герона для проверки, существует ли трапеция

double semiPerimeter1 = (c + d + baseDiff) / 2;

double semiPerimeter2 = (c + d + maxBase - minBase) / 2;

double triangleArea1 = sqrt(semiPerimeter1 \* (semiPerimeter1 - c) \* (semiPerimeter1 - d) \* (semiPerimeter1 - baseDiff));

double triangleArea2 = sqrt(semiPerimeter2 \* (semiPerimeter2 - c) \* (semiPerimeter2 - d) \* (semiPerimeter2 - (maxBase - minBase)));

double minHeight = 2 \* triangleArea1 / baseDiff;

double altMinHeight = 2 \* triangleArea2 / (maxBase - minBase);

if (h < minHeight || h < altMinHeight) {

cerr << "Ошибка: высота слишком мала для данных боковых сторон!\n";

return false;

}

return true;

}

};

int main() {

Trapezoid trap;

cout << "Введите длины оснований a и b: ";

cin >> trap.a >> trap.b;

if (trap.a <= 0 || trap.b <= 0) {

cerr << "Ошибка: длины оснований должны быть положительными!\n";

return 1;

}

cout << "Введите длины боковых сторон c и d: ";

cin >> trap.c >> trap.d;

if (trap.c <= 0 || trap.d <= 0) {

cerr << "Ошибка: длины боковых сторон должны быть положительными!\n";

return 1;

}

cout << "Введите высоту h: ";

cin >> trap.h;

if (trap.h <= 0) {

cerr << "Ошибка: высота должна быть положительной!\n";

return 1;

}

// Проверяем, существует ли такая трапеция

if (!trap.isValid()) {

return 1;

}

cout << fixed << setprecision(2);

cout << "Средняя линия: " << trap.middleLine() << endl;

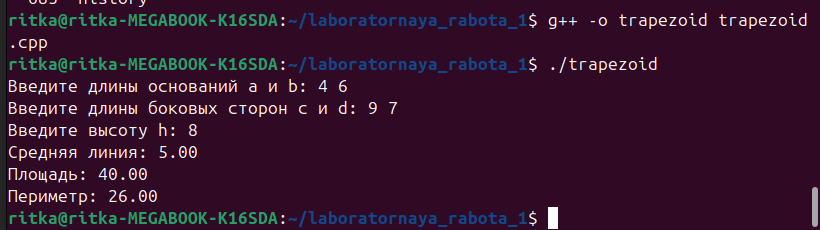
cout << "Площадь: " << trap.area() << endl;

cout << "Периметр: " << trap.perimeter() << endl;

return 0;

}

**Результат работы программы:**

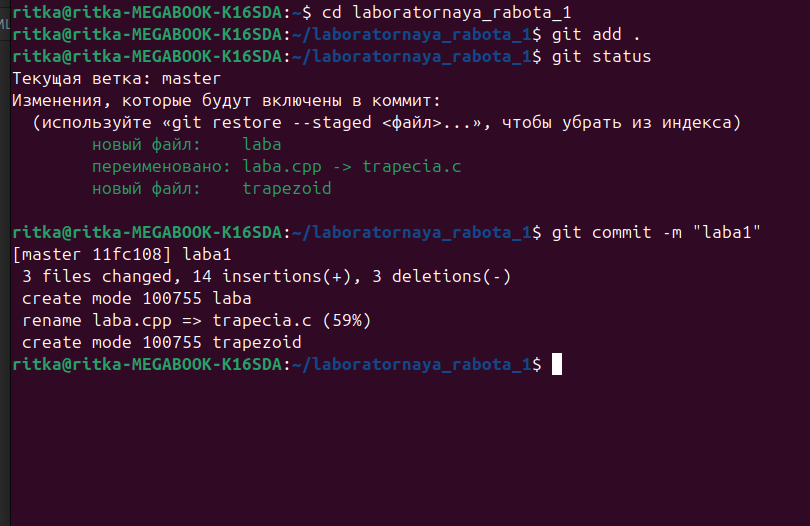


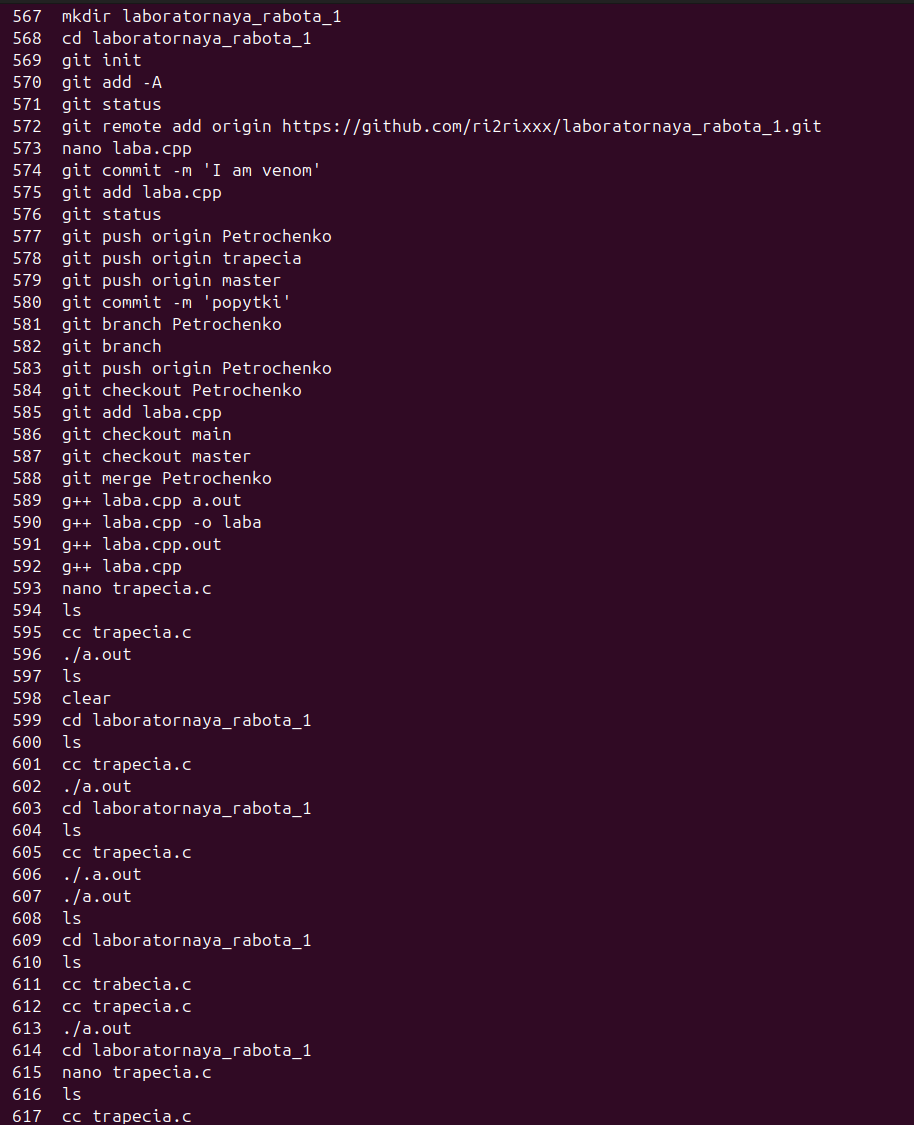
**Полное описание всех этапов процесса работы:**

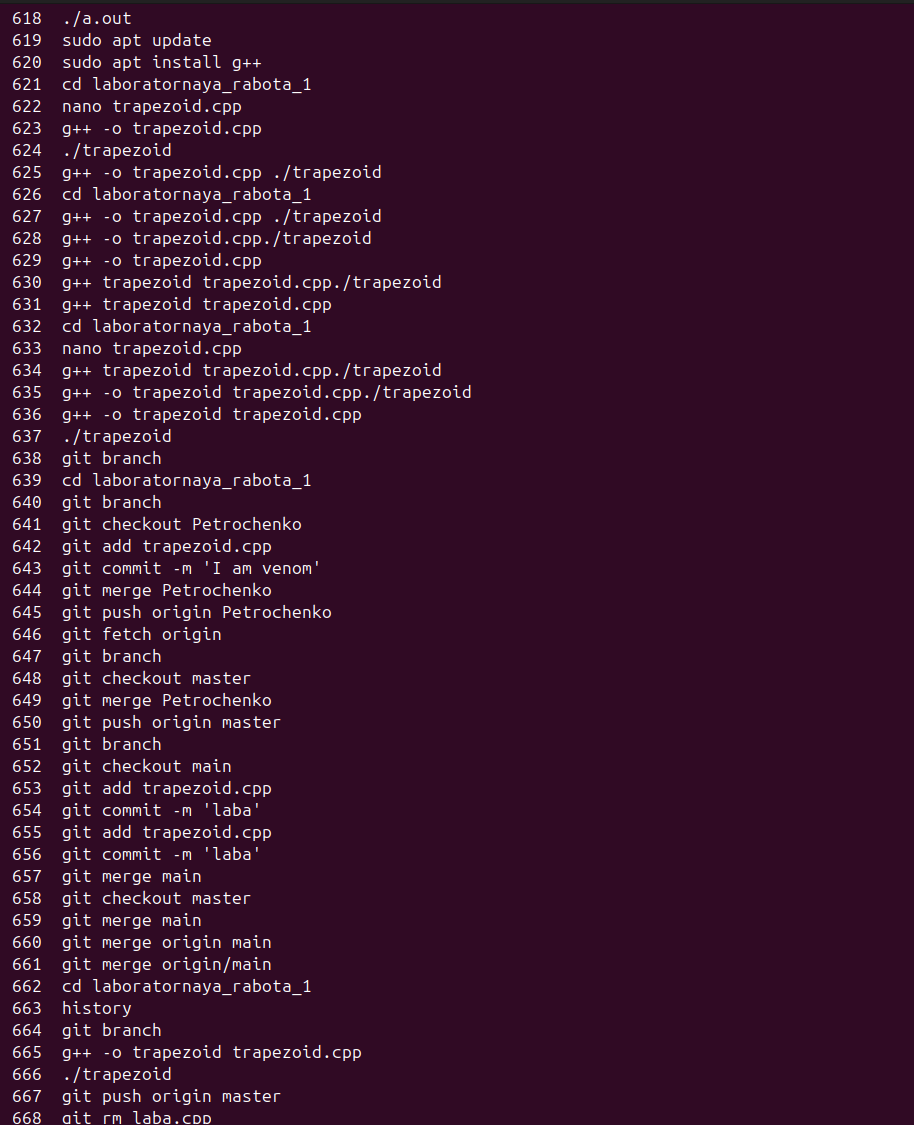
1. Создание пустого локального репозитория “laboratornaya\_rabota\_1” в GitHub
2. Создание файла README.md в основной ветке “main”.
3. Создание папки “laboratornaya\_rabota\_1” с будущим кодом и его объявление
4. Привязка Гитхаба с Гитом
5. Создание репозитория и ветки “master”

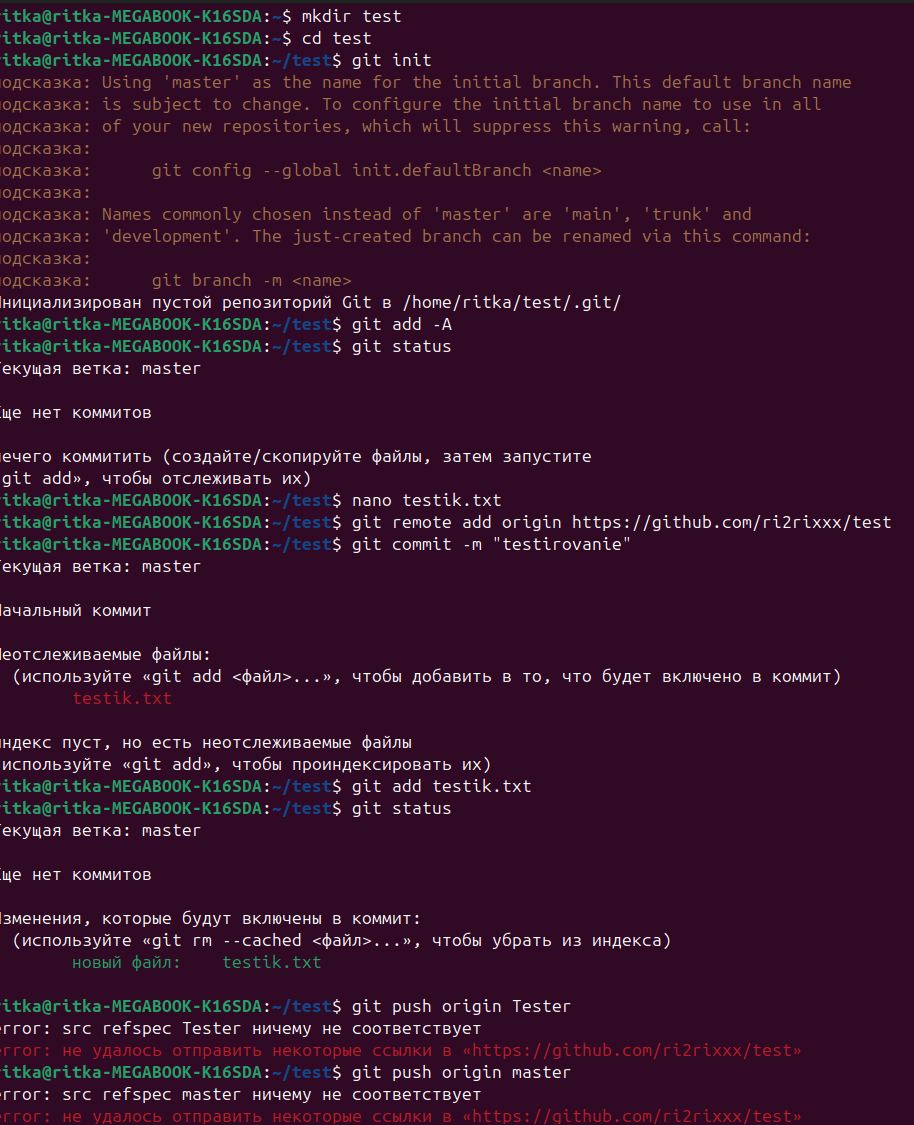


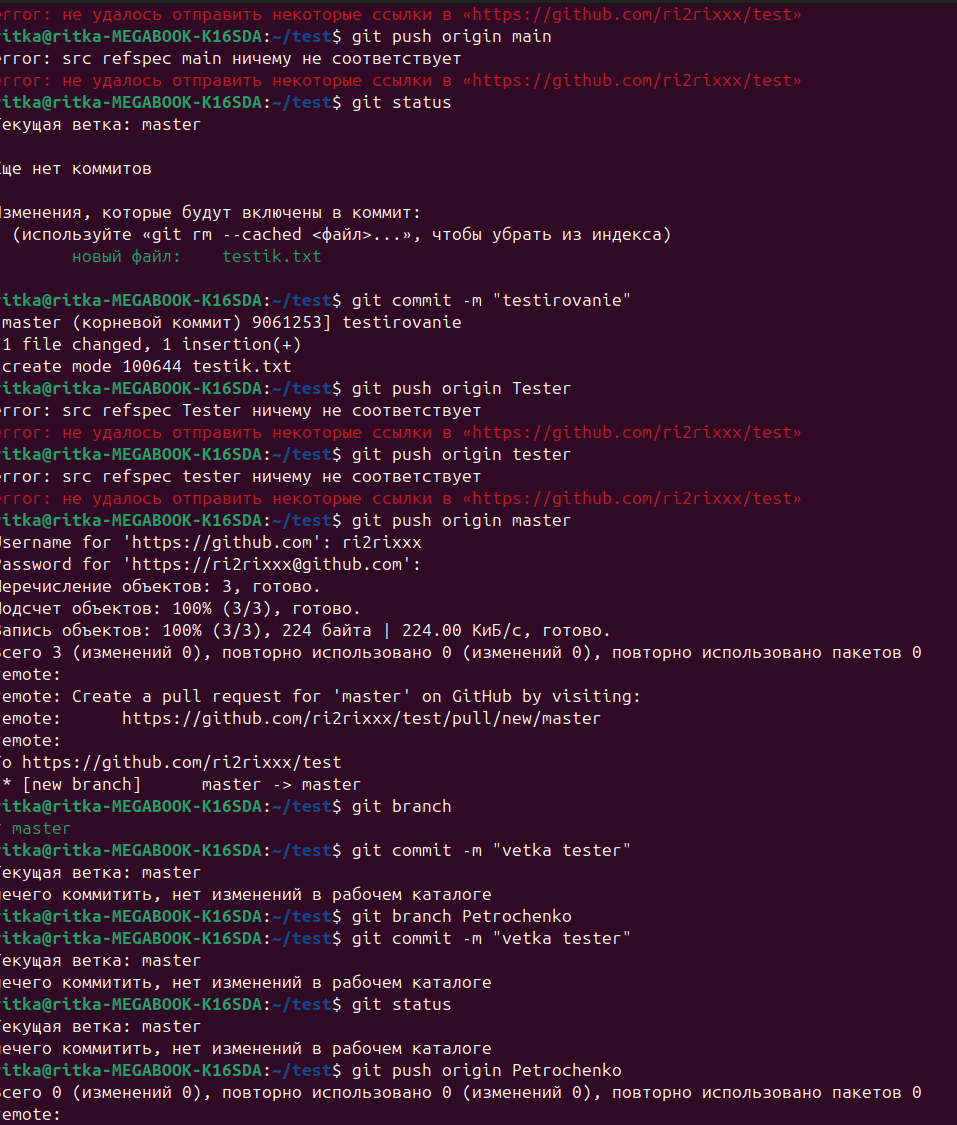
1. Оформление кода, доработка кода, сохранение и отладка
2. Добавление кода в папку
3. Создание коммита
4. Связывание репозитория с локальным репозиторием
5. Отображение файла на сайте в ветке “master”
6. Создание отдельной ветки “Petrochenko” с файлом trapezoid.cpp в репозиторий
7. Создание коммита
8. Отображение файла на сайте в ветке “Petrochenko”
9. Выполнение слияния дополнительной ветки “Petrochenko” в основную “main”. Сохранение и загрузка в репозиторий

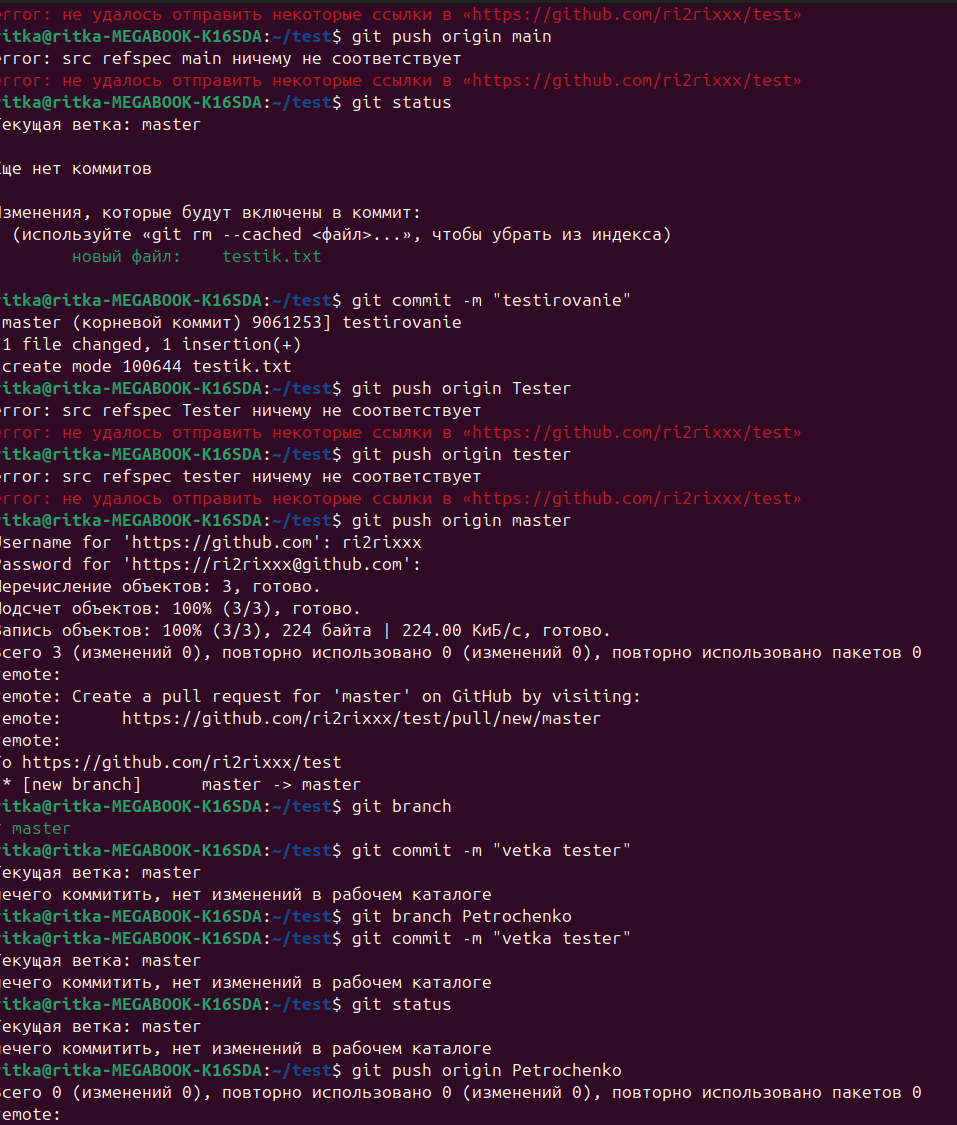












**Вывод**

В результате работы были выполнены все поставленные цели и задачи: изучена платформа GitHub и практическое использование Git для хранения, обновления и распространения исходного кода проекта.