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 №1

ПО КУРСУ «ОСНОВЫ ПРОГРАММНОЙ ИНЖЕНЕРИИ»

«Работа с Git и GitHub»

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

Астафьев Алексей Алексеевич

Проверила: ассистент ВШ КЦТ

Балабасова Алиса Дмитриевна

Хабаровск 2025 г.

Цель: Освоить создание репозитория, добавление файлов, работу с ветками, форками и pull request.

Задания:

1. Создать репозиторий на GitHub, добавить описание проекта.

2. Настроить gitignore, README.md (у себя локально). Закоммитить и отправить изменения в репозиторий (консольно, не через интерфейс GitHub).

3. Добавить бейдж статуса сборки в README.md.

4. Настроить SSH-ключ для аутентификации с GitHub.

5. Создать новую ветку "feature-branch" (у себя локально).

6. Внести изменения в код, закоммитить. Переключиться на основную ветку, объединить изменения. Разрешить возможные конфликты при слиянии.

7. Сделать форк существующего репозитория или своего же проекта.

8. Внести изменения в код, закоммитить и запушить с помощью ssh в форкнутый репозиторий.

9. Создать Pull Request между своим форком и оригинальным репозиторием (или между

разными ветками в своём репозитории). Провести code review изменённого кода перед мержем, используя GitHub review tools.

Ход работы:

1. Создать репозиторий на GitHub, добавить описание проекта.

С помощью MS Visual Studio создаём новое консольное приложение. После этого создаём репозиторий и добавляем описание, как показано на рисунке 1.

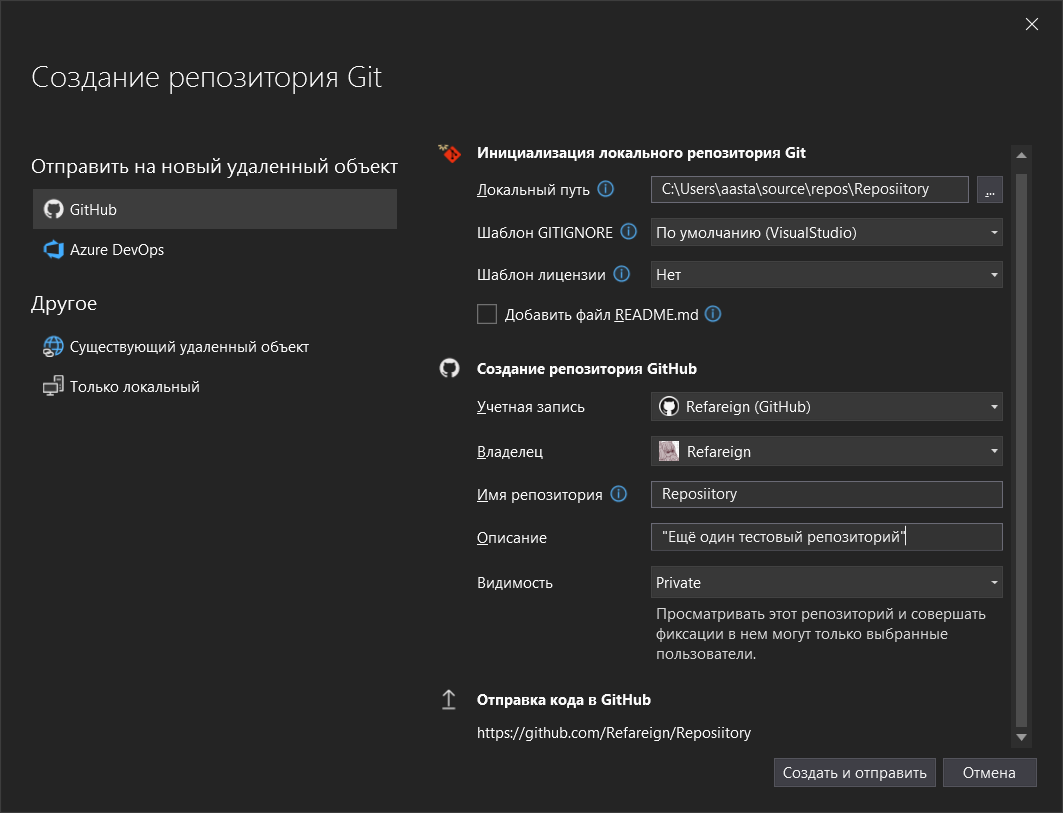


Рисунок 1 – создание репозитория.

Как видно на рисунке 2, репозиторий был успешно создан с описанием проекта.

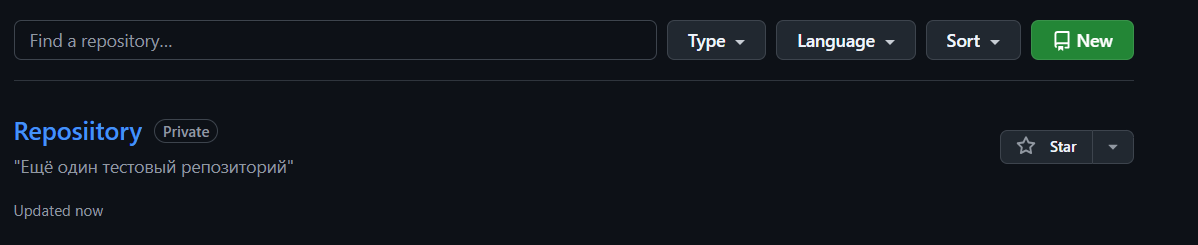


Рисунок 2 – созданный репозиторий в GitHub.

2. Настроить gitignore, README.md (у себя локально). Закоммитить и отправить изменения в репозиторий (консольно, не через интерфейс GitHub).

Так как мой репозиторий был создан с помощью MS Visual Studio, то файл .gitignore автоматически появился в корневой папке проекта, и настраивать его, как правило, не нужно. Однако, стоит пояснить, что содержимое этого файла, показанное на рисунке 3, используется для того, чтобы указать Git, как файлы отслеживать НЕ надо. Например, файлы сборки.

Git будет игнорировать все файлы и папки, перечисленные в .gitignore, и они не будут добавлены в коммиты

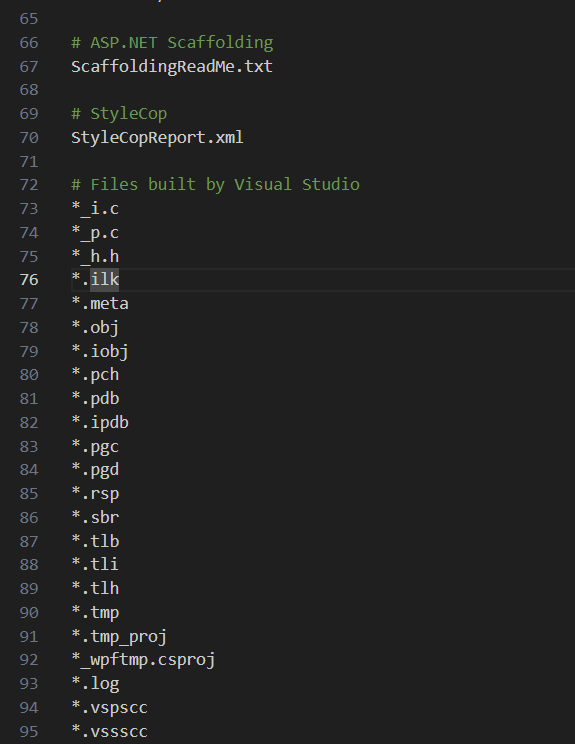


Рисунок 3 – файл .gitignore

Файл README.md автоматически не создаётся, если не выбрать его автоматическое создание. Поэтому открываем консоль и переходим в корневую папку проекта, где создаём файл, как показано на рисунке 4.

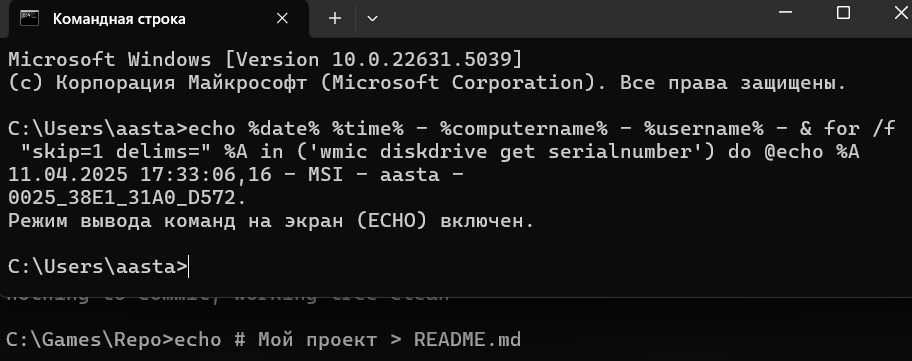


Рисунок 4 – создание README.md

После чего этот файл можно редактировать любым текстовым редактором. Пишем, что нашей душе угодно и коммитим в проект.

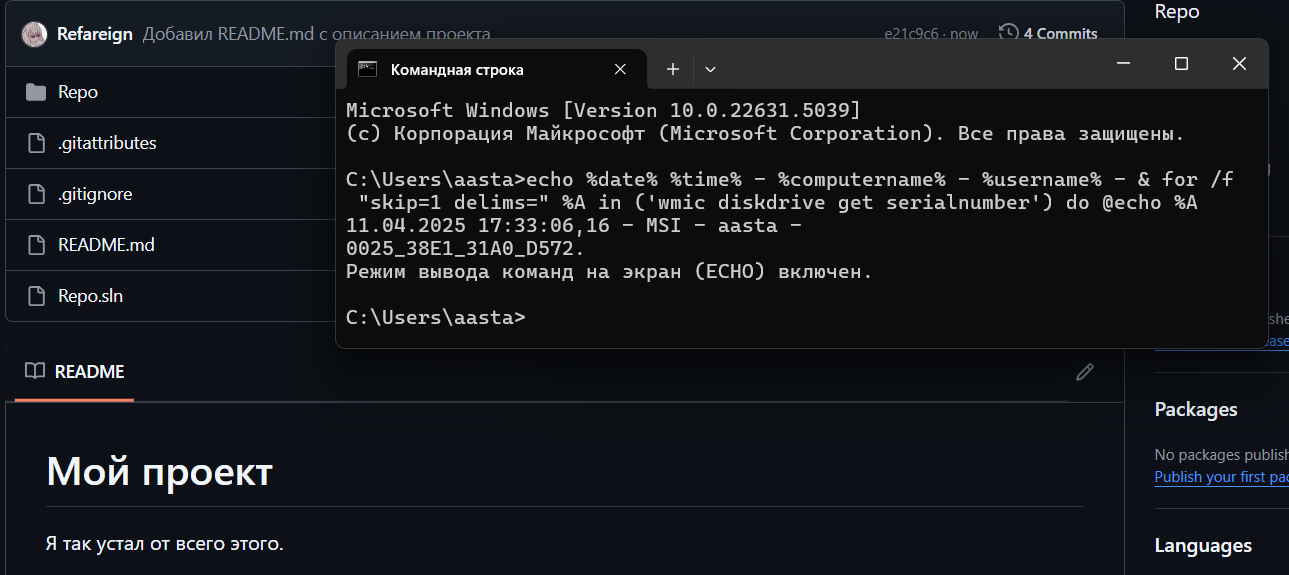


Рисунок 5 – файл успешно отправлен в репозиторий

3. Добавить бейдж статуса сборки в README.md.

Чтобы добавит бейдж статуса сборки, в проекте должен быть workflow-файл. Создадим такой, например, для сборки проекта проекта в Visual Studio на Windows по пути .github/workflows/ с помощью интерфейса GitHub.

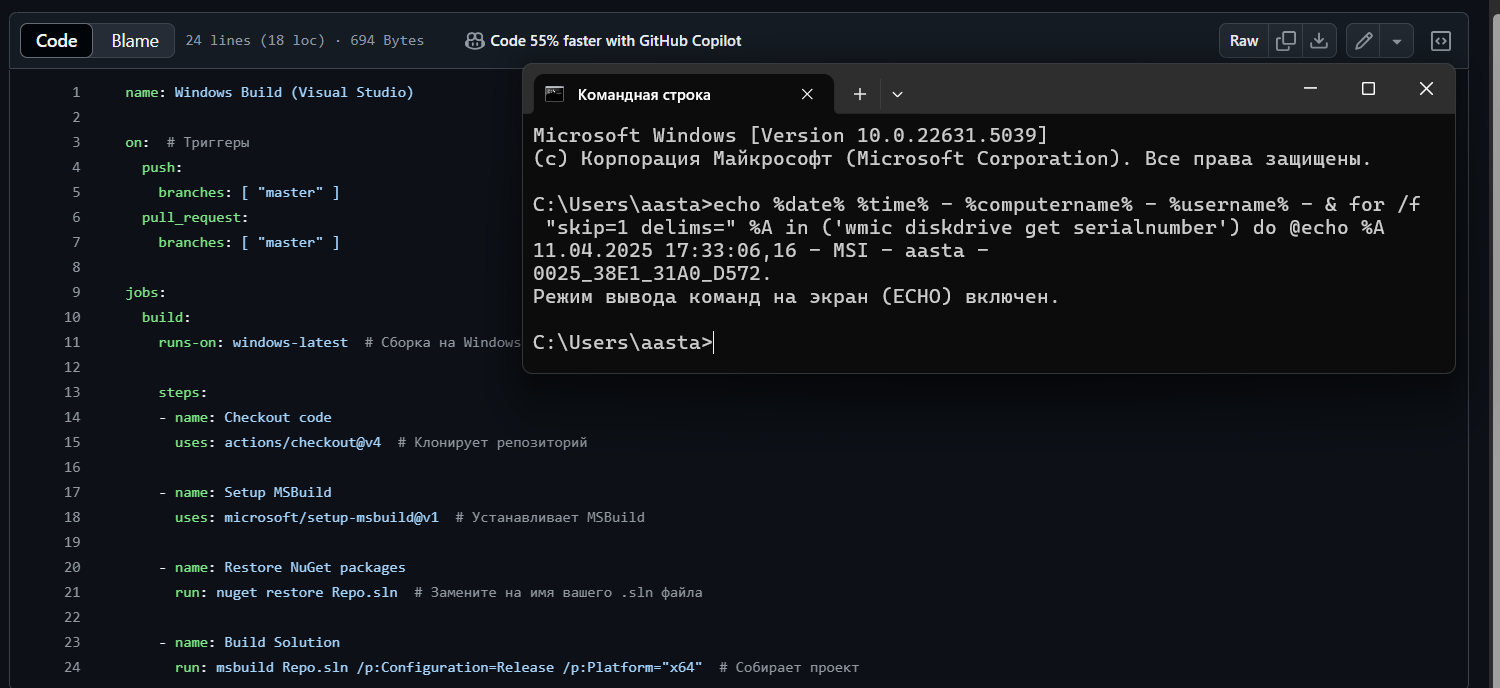


Рисунок 6 – созданный файл.

После этого запустится workflow, к которому можно будет прикрепить бейдж. Интерфейс GitHub предложит нам скопировать markdown код, который мы и вставим в README.md.

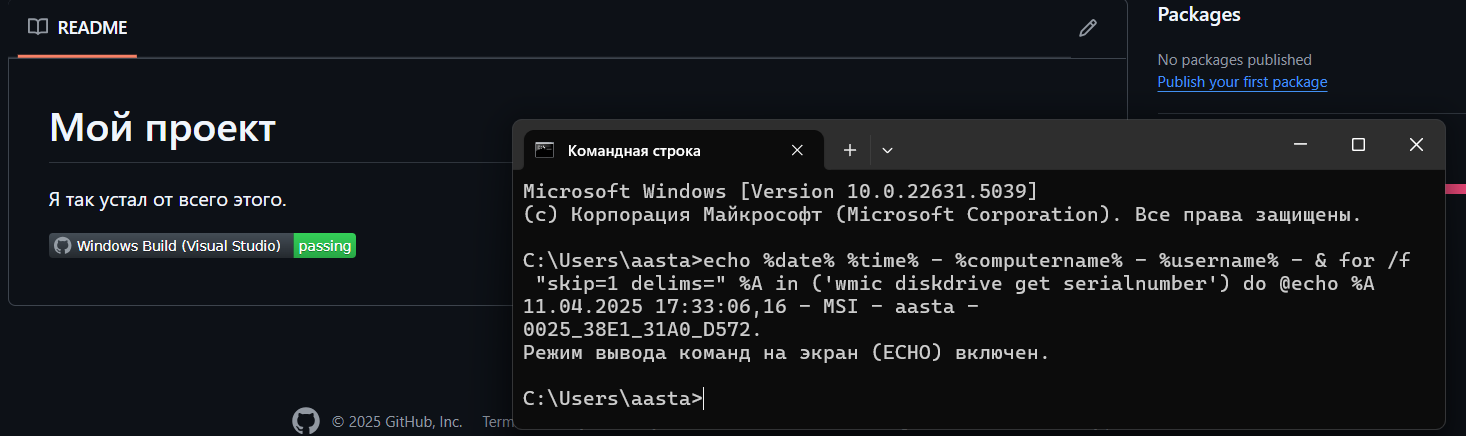


Рисунок 7 – успешно добавленный бедж статуса сборки

4. Настроить SSH-ключ для аутентификации с GitHub.

Ключ успешно добавлен в интерфейсе GitHub, после чего я успешно прошёл аутентификацию в консоли.

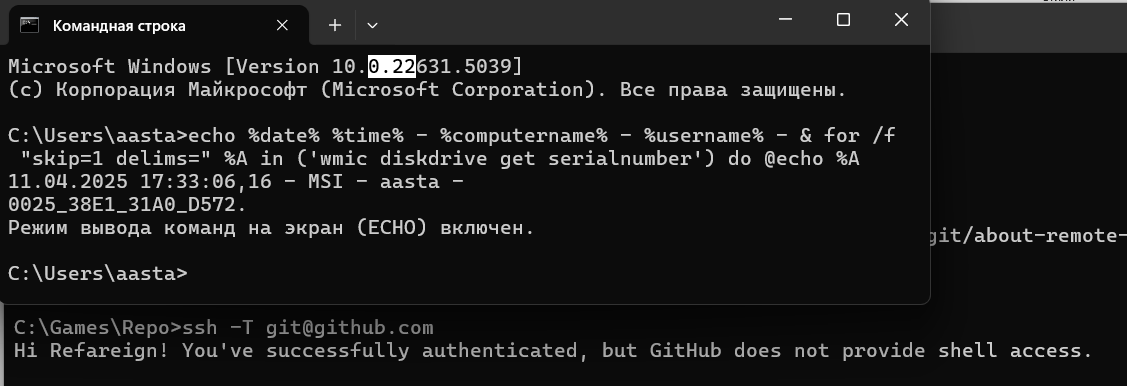


Рисунок 8 – аутентификация с GitHub

5. Создать новую ветку "feature-branch" (у себя локально).

С помощью всего одной консольной команды создаём новую ветку и автоматически на неё переключаемся.

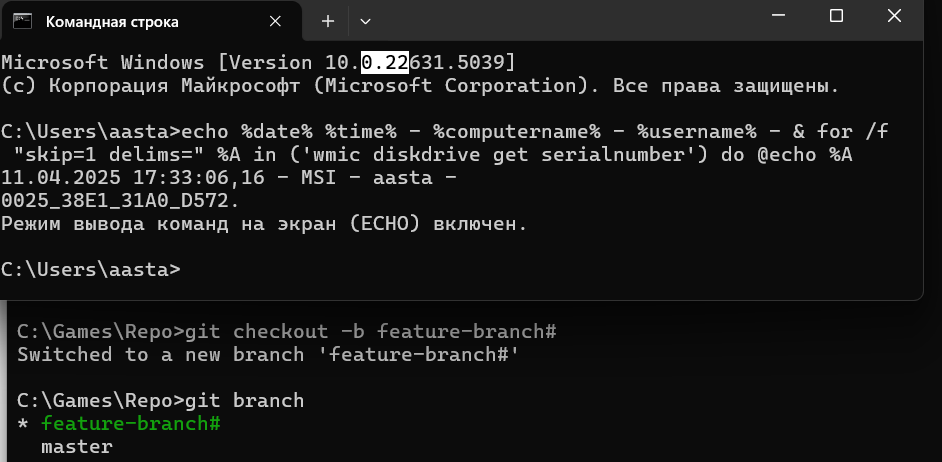


Рисунок 9 – создание новой ветки.

6. Внести изменения в код, закоммитить. Переключиться на основную ветку, объединить изменения. Разрешить возможные конфликты при слиянии.

Все эти операции успешно выполнены, никаких конфликтов не возникло. Однако, если бы конфликты всё-таки возникли, то Git бы пометил конфликтующие участки кода, и надо было бы выбрать, какую версию кода использовать.

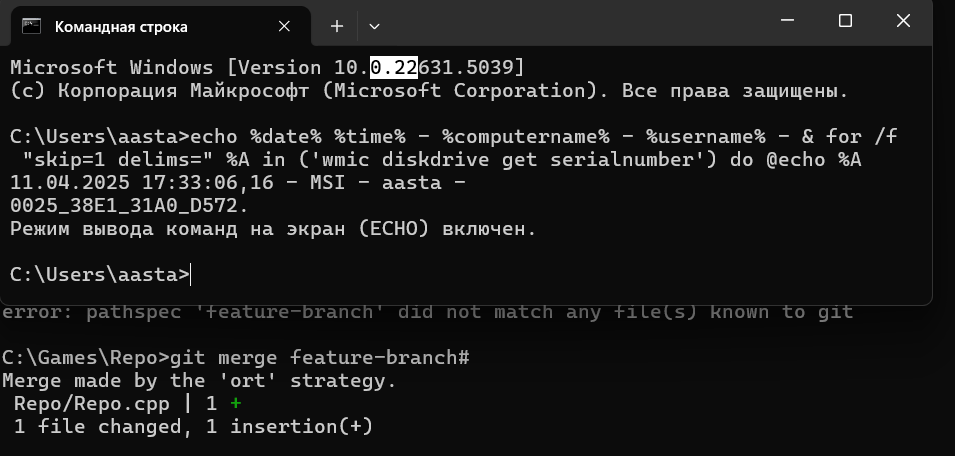


Рисунок 10 – успешное слияние веток

7. Сделать форк существующего репозитория или своего же проекта.

Был успешно сделан форк случайно выбранного репозитория The Algorithms.

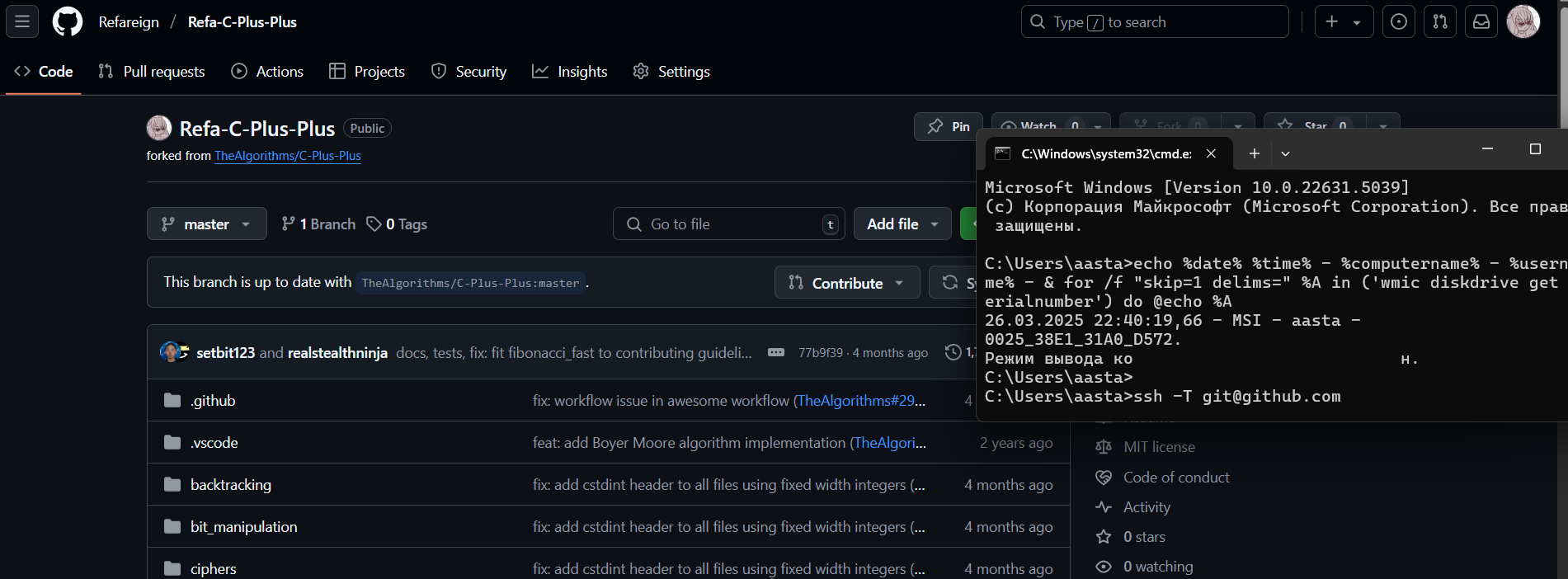


Рисунок 10 – форк репозитория The Algorithms

8. Внести изменения в код, закоммитить и запушить с помощью ssh в форкнутый репозиторий.

Все эти действия успешно сделаны.

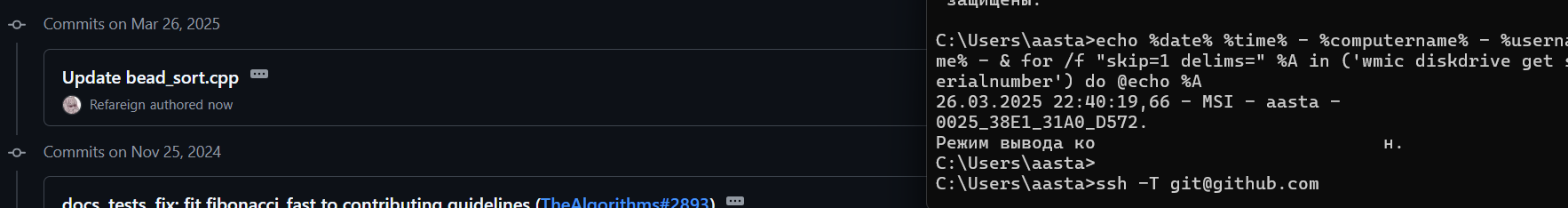


Рисунок 11 – коммит форкнутого репозитория

9. Создать Pull Request между своим форком и оригинальным репозиторием (или между разными ветками в своём репозитории). Провести code review изменённого кода перед мержем, используя GitHub review tools.

етками в своём репозитории.

Pull Request успешно создан (и так же успешно закрыт от греха подальше). Так как оригинальный репозиторий мне не принадлежит, то мержеить я их не могу (да и не надо).

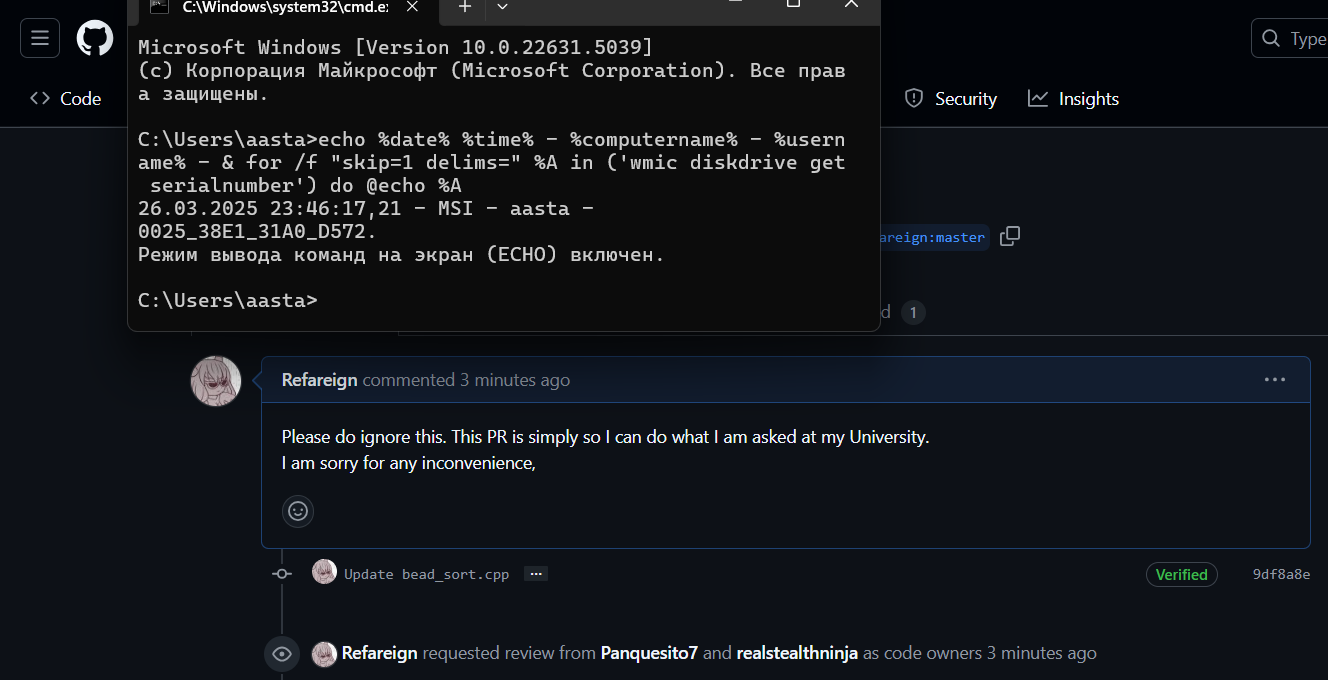


Рисунок 12 - Pull Request.

С помощью GitHub review tools прекрасно видно, как много ценных и важных изменений в коде я сделал.

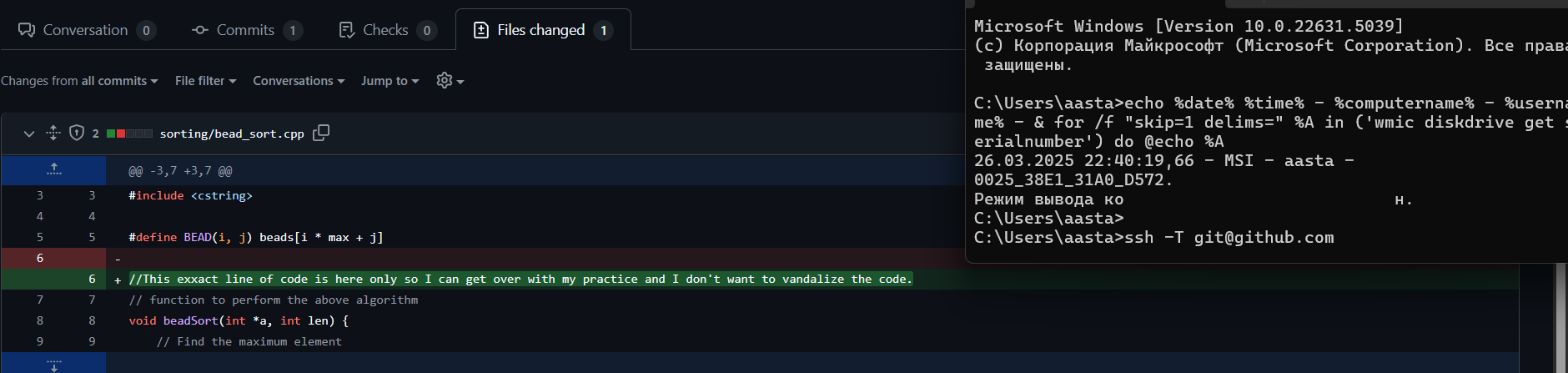


Рисунок 13 – просмотр изменённого кода.

Вывод: >:(