Отчёт по лабораторной работе №1

Дисциплина: Математическое моделирование

Боровиков Даниил Александрович НПИбд-01-22

Содержание

# 1 Цель работы

Приобрести практические навыки работы с git и разобрать основные его особенности.

# 2 Выполнение лабораторной работы

Для начала установим параметры окончаний строк (рис. 1).

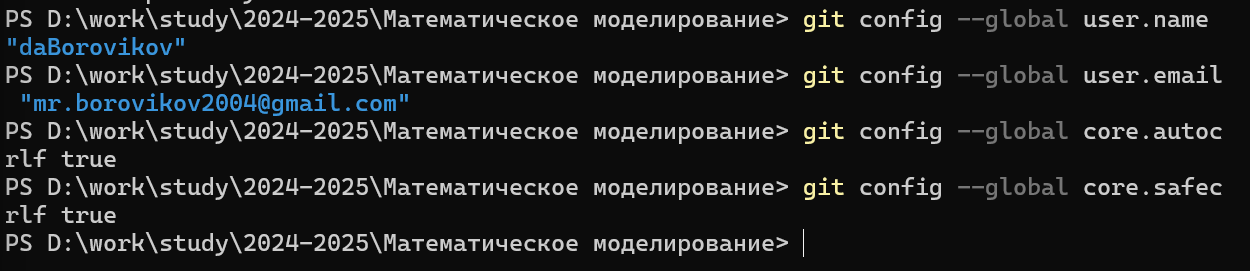


Рис. 1: Параметры установки окончаний строк

По умолчанию, git будет печатать не-ASCII символов в именах файлов в виде восьмеричных последовательностей. Что бы избежать нечитаемых строк, установим соответствующий флаг (рис. 2).

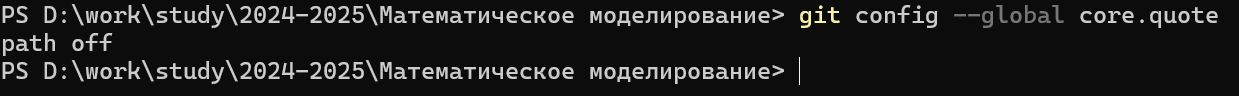


Рис. 2: Установка отображения unicode

Начнём работу в пустом рабочем каталоге с создания пустого каталога с именем hello, затем войдём в него и создадим там файл с именем hello.html (рис. 3).

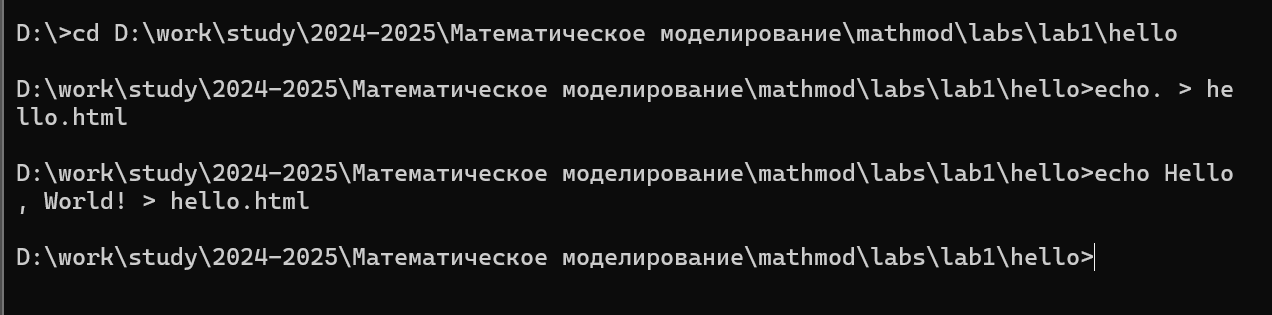


Рис. 3: Создание страницы «Hello, World»

Чтобы создать git репозиторий из этого каталога, выполним команду git init (рис. 4).

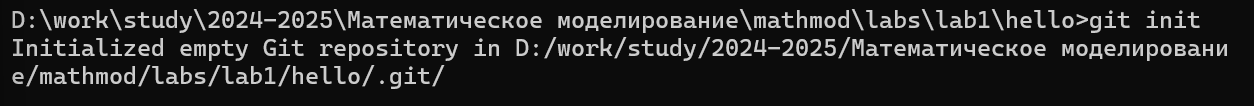


Рис. 4: Создание репозитория

Добавим файл в репозиторий (рис. 5).

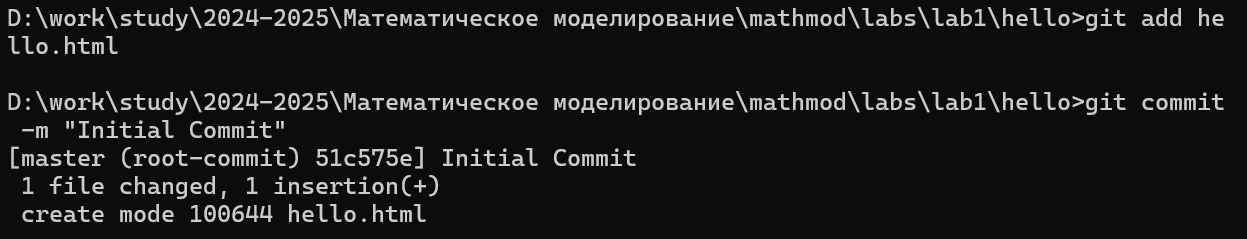


Рис. 5: Добавление файла в репозиторий

Используем команду git status, чтобы проверить текущее состояние репозитория (рис. 6).

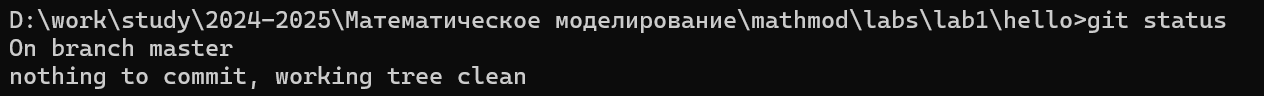


Рис. 6: Проверка состояние репозитория

Добавим кое-какие HTML-теги к нашему приветствию. Изменим содержимое файла hello.html (рис. 7).

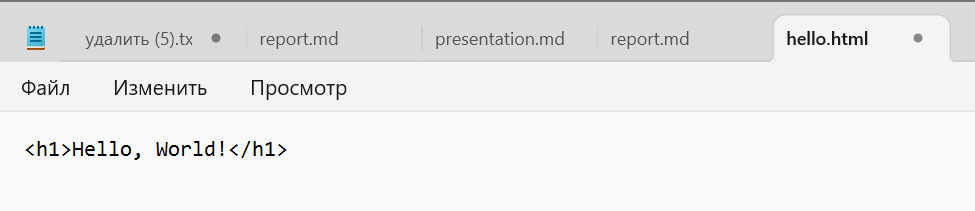


Рис. 7: Изменение содержимого файла hello.html

Проверим состояние рабочего каталога (рис. 8).

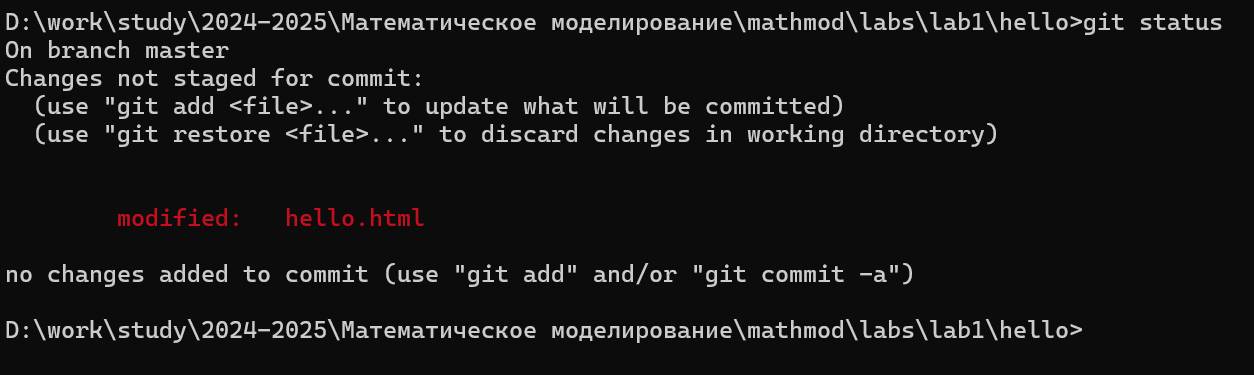


Рис. 8: Проверка состояния рабочего каталога

Теперь выполним команду git, чтобы проиндексировать изменения. Проверим состояние (рис. 9).

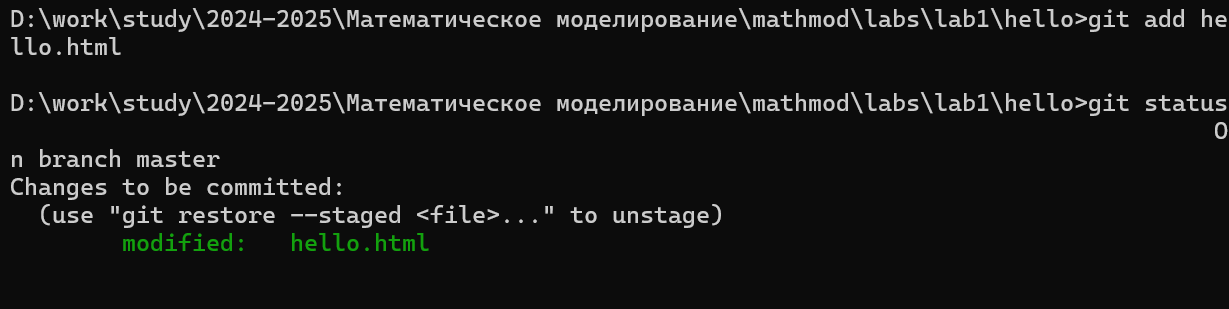


Рис. 9: Индексация изменений

Сделаем коммит и проверим состояние. Откроется редактор. В первой строке введём комментарий: «Added h1 tag». Сохраним файл и выйдем из редактора (рис. 10).

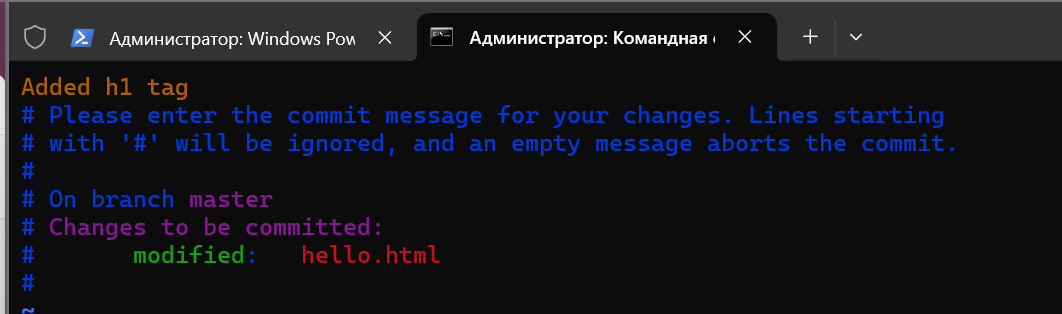


Рис. 10: Добавление комментария, сохранение файла и выход из редактора

Теперь еще раз проверим состояние (рис. 11).

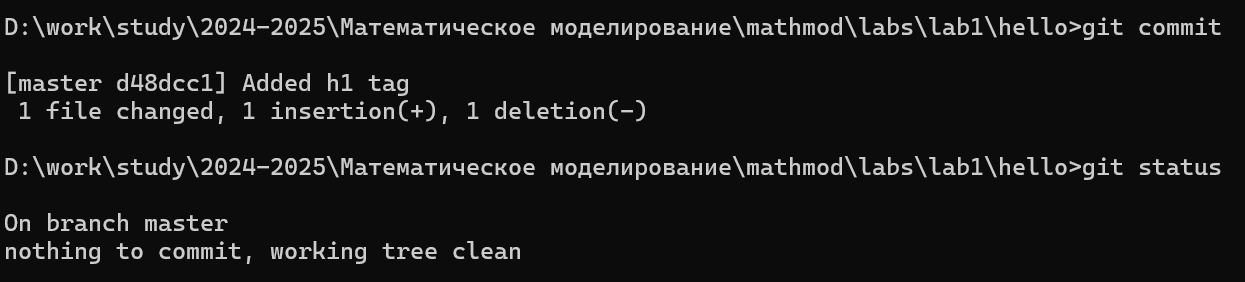


Рис. 11: Повторная проверка состояния

Изменим страницу «Hello, World», чтобы она содержала стандартные теги

и

(рис. 12).

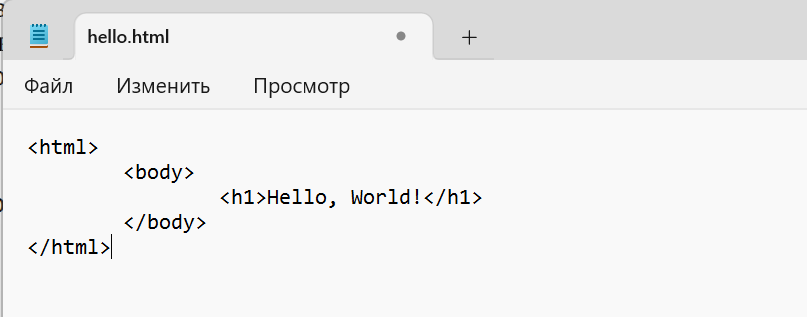


Рис. 12: Изменение страницы «Hello, World»

Теперь добавим это изменение в индекс git (рис. 13).

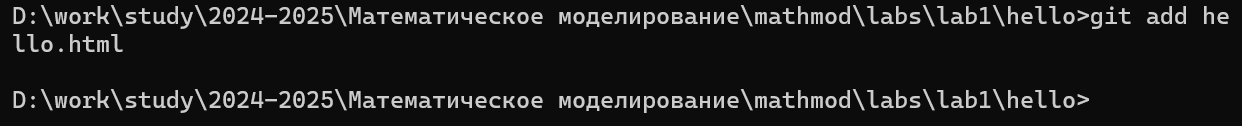


Рис. 13: Добавление изменения в индекс git

Теперь добавим заголовки HTML (секцию

) к странице «Hello, World» (рис. 14).

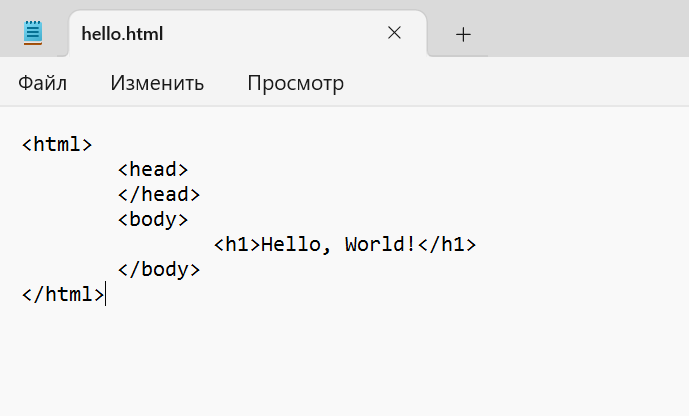


Рис. 14: Добавление заголовков HTML к странице «Hello, World»

Проверим текущий статус. Произведём коммит проиндексированного изменения (значение по умолчанию), а затем еще раз проверим состояние. Далее добавим второе изменение в индекс, а затем проверим состояние с помощью команды git status и сделаем коммит второго изменения (рис. 15).

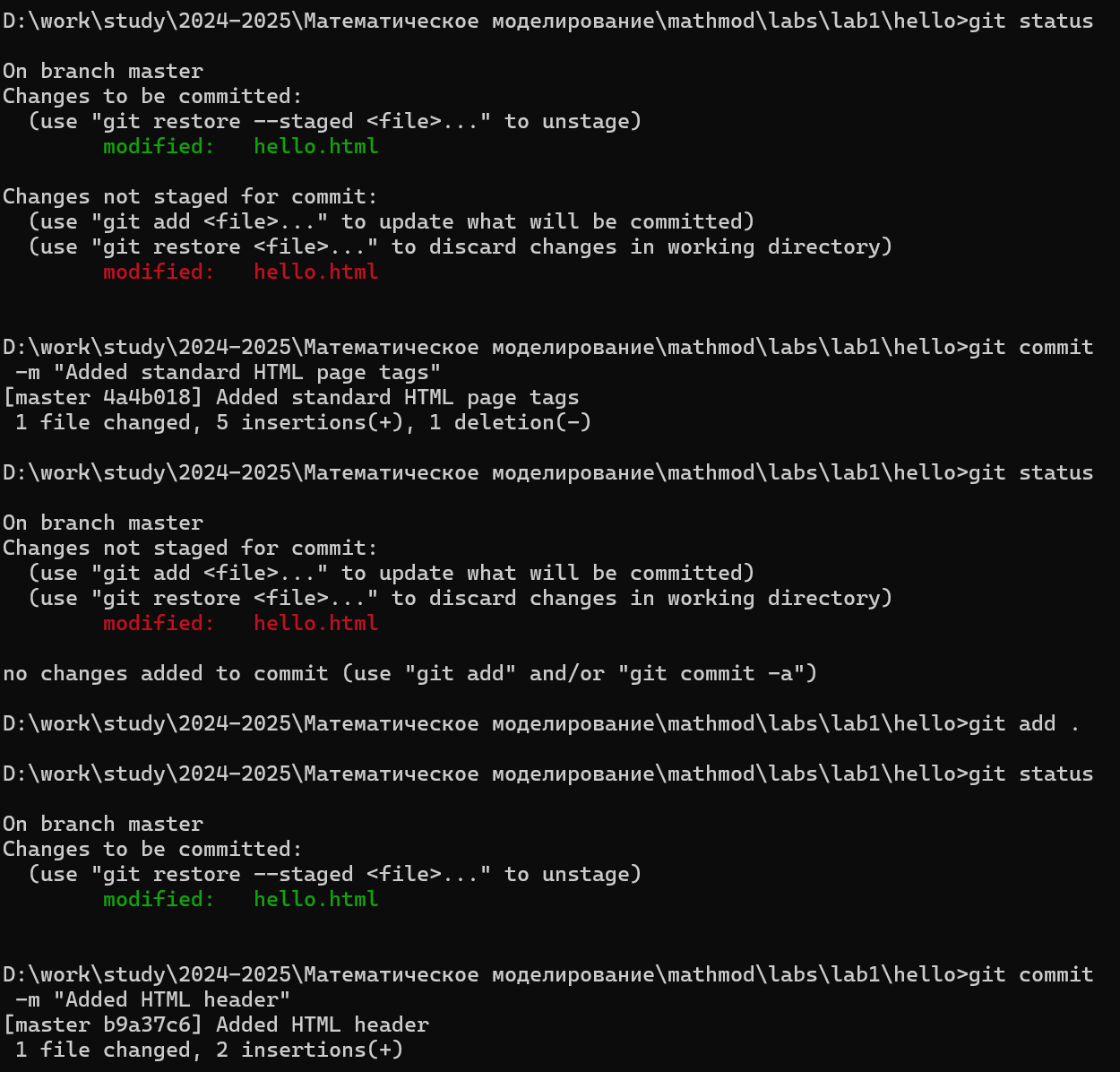


Рис. 15: Добавление стандартных тегов страницы

Получим список произведенных изменений различными способами (рис. 16).

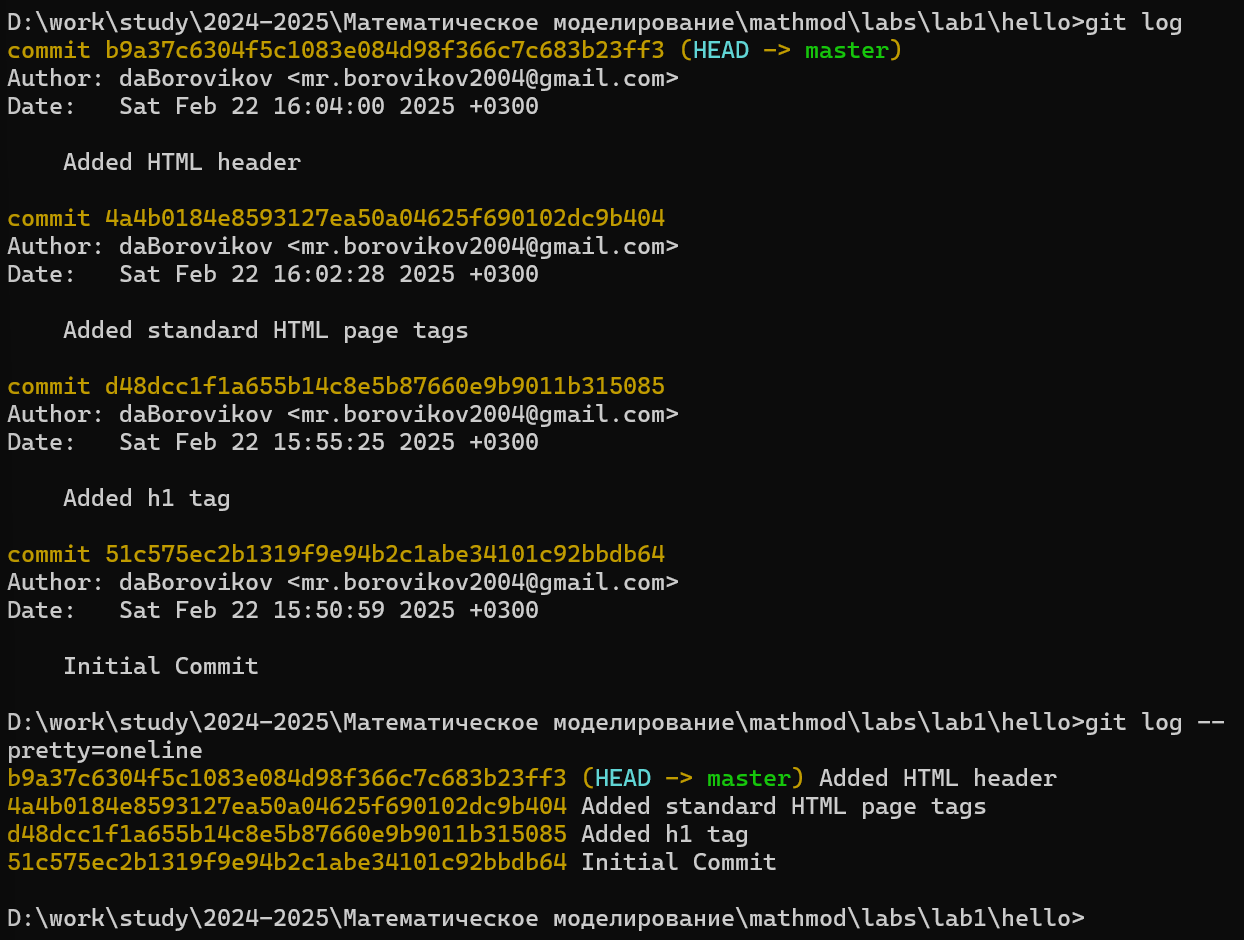


Рис. 16: История

Получим хэши предыдущих версий. Затем проверим содержимое файла hello.html и вернёмся к последней версии в ветке master (рис. 17).



Рис. 17: Получение старых версий

Создадим тег первой версии. Далее переключимся на предыдущую версию. Это версия c тегами

и

, но еще пока без

. Давайте сделаем ее версией v1-beta (рис. 18).

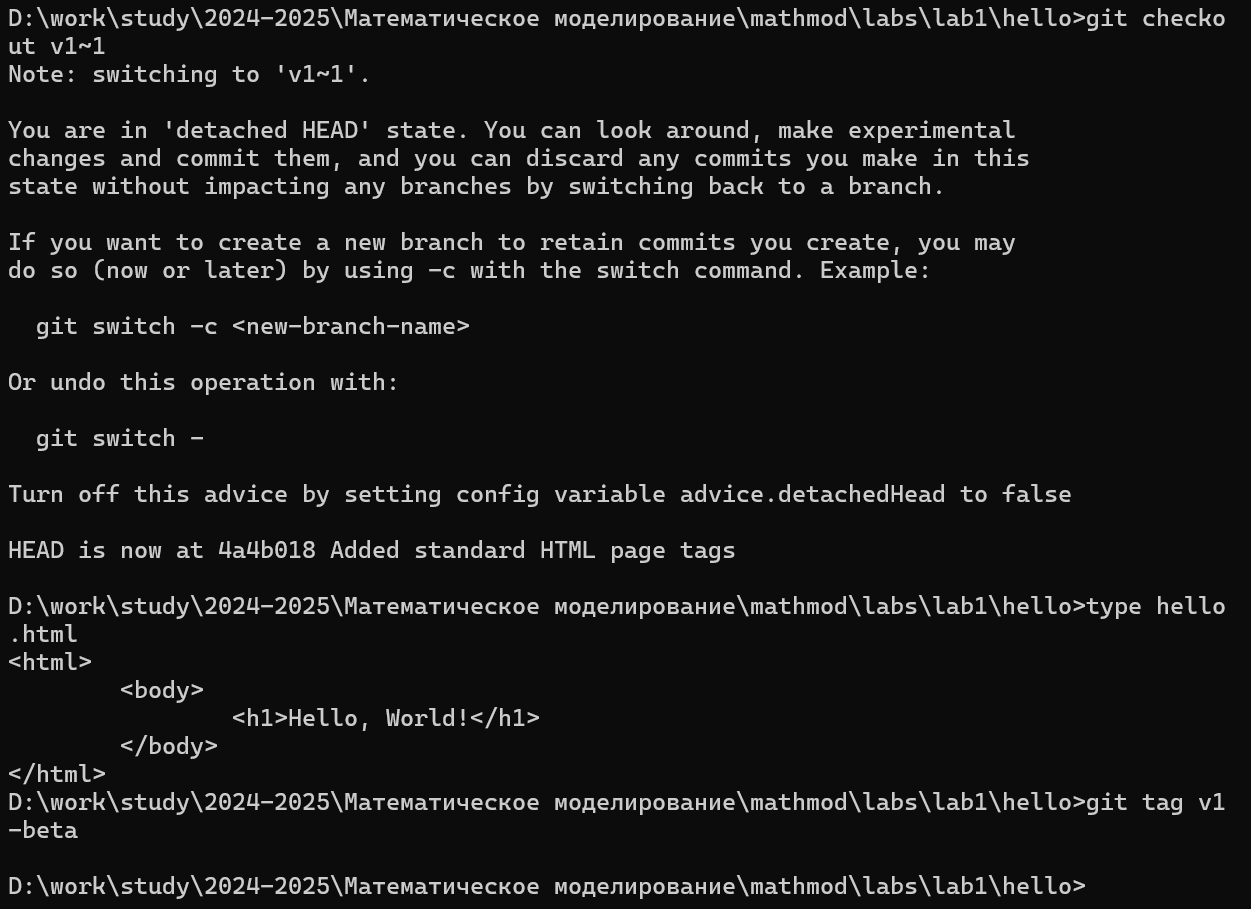


Рис. 18: Создание тегов версий

Теперь попробуем попереключаться между двумя отмеченными версиями (рис. 19).

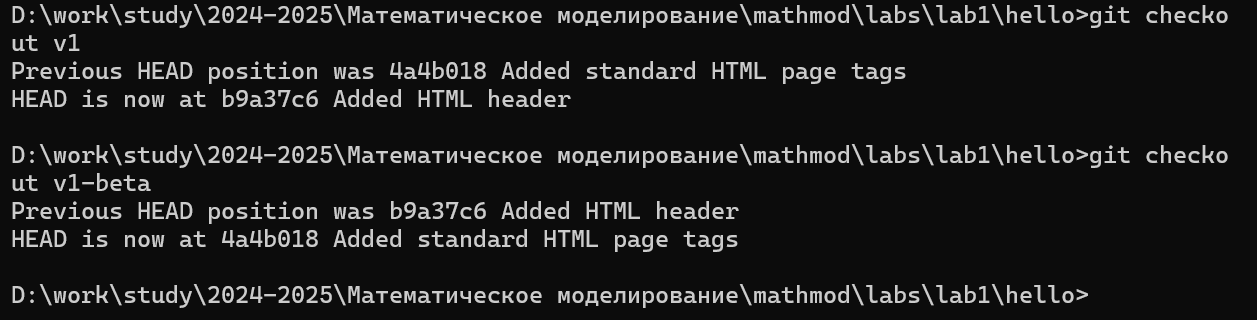


Рис. 19: Переключение по имени тега

Мы можем увидеть, какие теги доступны, используя команду git tag и также можем посмотреть теги в логе (рис. 20).

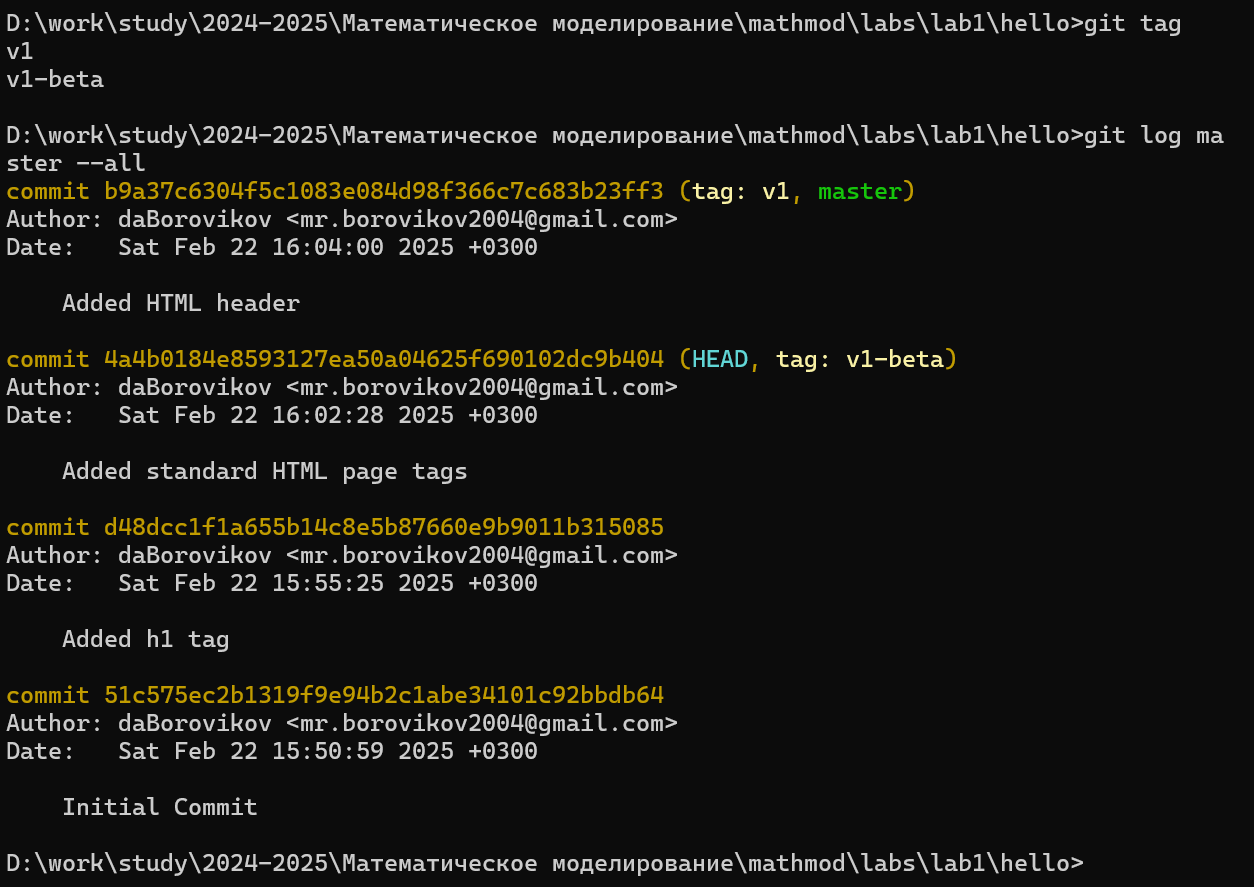


Рис. 20: Просмотр тегов с помощью команды tag

Убедимся, что мы находимся на последнем коммите ветки master, прежде чем продолжить работу (рис. 21).

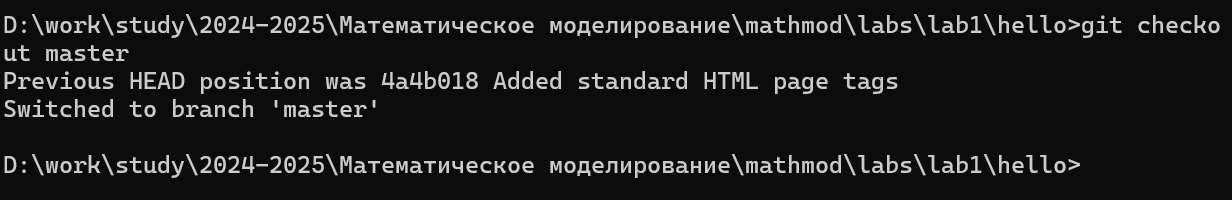


Рис. 21: Переключитесь на ветку master

Внесём изменение в файл hello.html в виде нежелательного комментария (рис. 22).

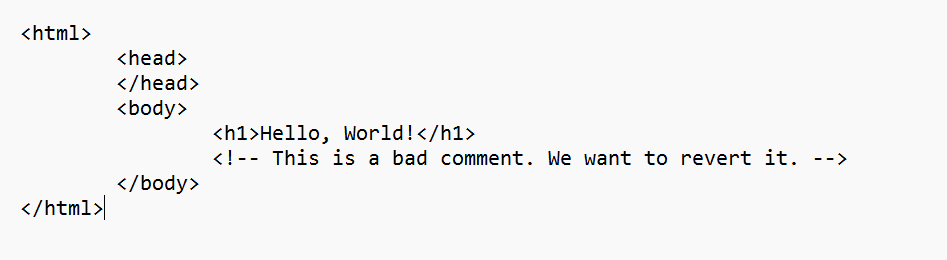


Рис. 22: Изменение hello.html

Сначала проверим состояние рабочего каталога (рис. 23).

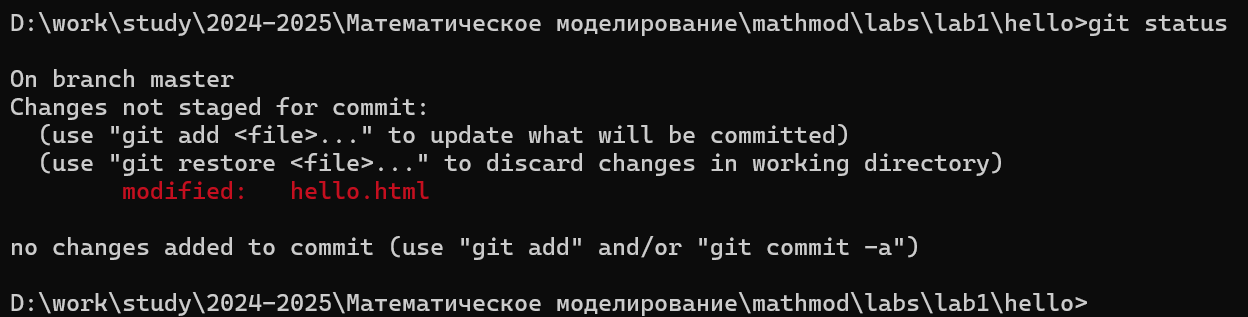


Рис. 23: Проверка состояния

Используем команду git checkout для переключения версии файла hello.html в репозитории (рис. 24).

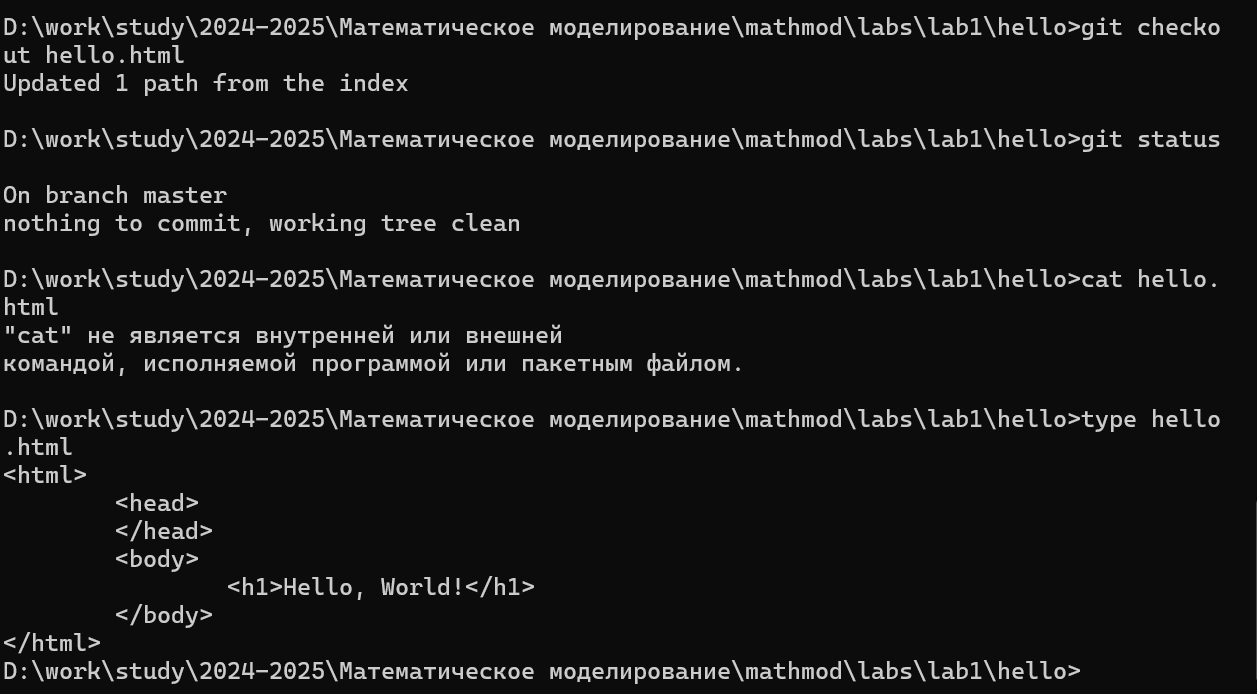


Рис. 24: Отмена изменений в рабочем каталоге

Внесём изменение в файл hello.html в виде нежелательного комментария (рис. 25).

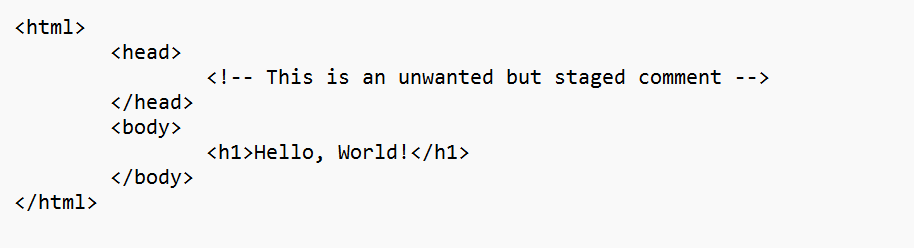


Рис. 25: Внесение изменения в файл hello.html

Проиндексируем это изменение (рис. 26).

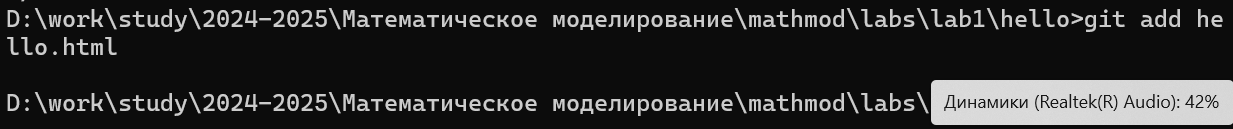


Рис. 26: Индексация изменения

Проверим состояние нежелательного изменения (рис. 27).

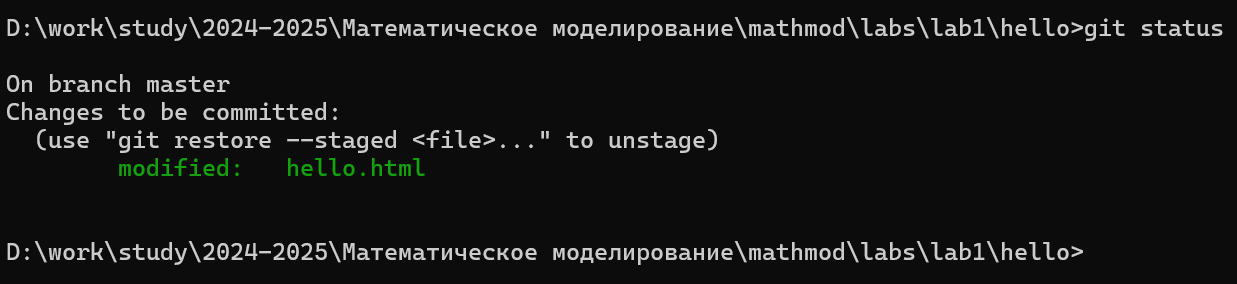


Рис. 27: Проверка состояния

К счастью, вывод состояния показывает нам именно то, что мы должны сделать для отмены индексации изменения (рис. 28).

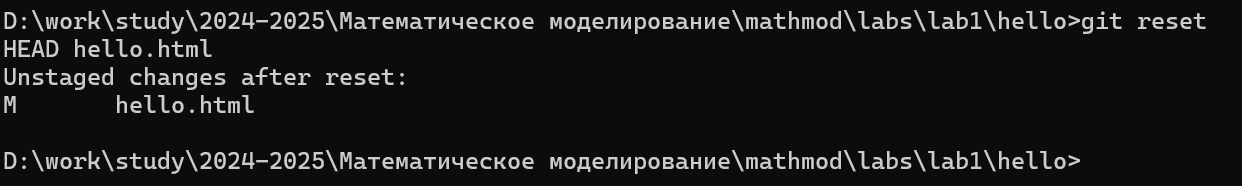


Рис. 28: Выполнение сброса буферной зоны

Переключимся на версию коммита (рис. 29).

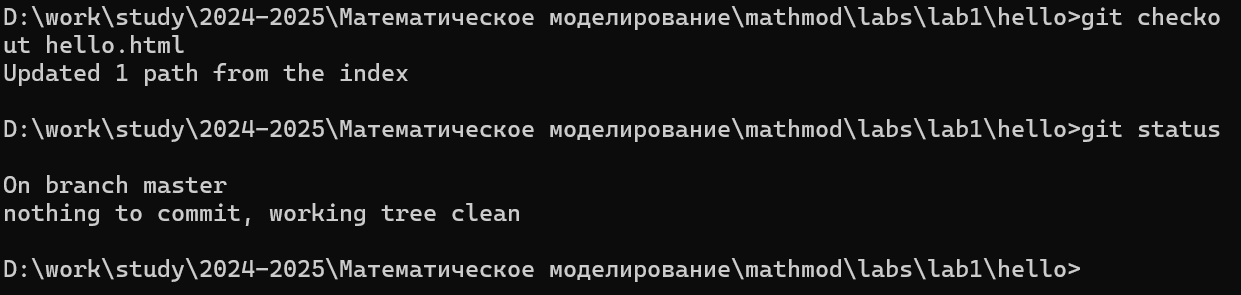


Рис. 29: Переключение на версию коммита

Изменим файл hello.html (рис. 30).

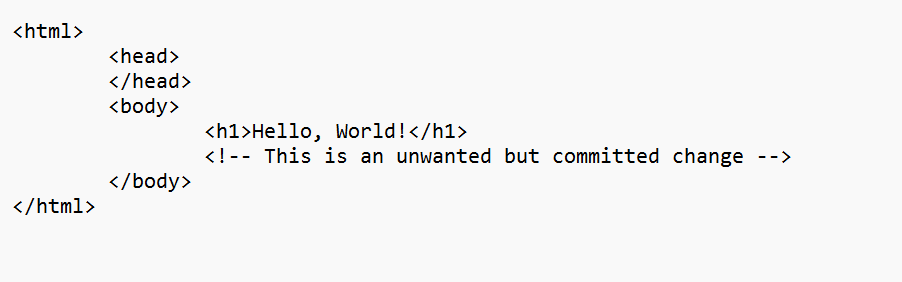


Рис. 30: Изменение файла hello.html

Далее сделаем коммит (рис. 31).

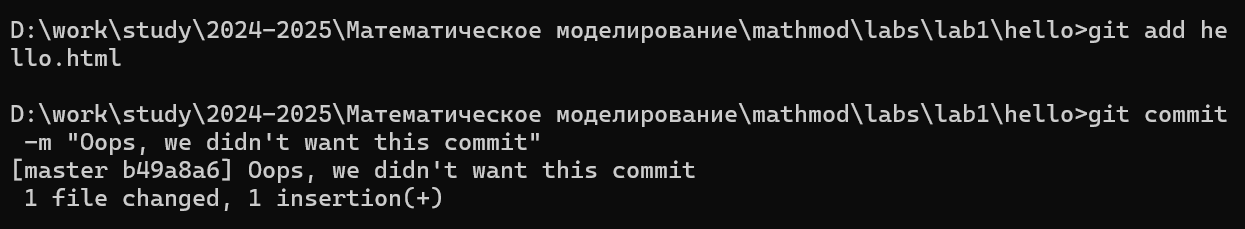


Рис. 31: Коммит

Чтобы отменить коммит, нам необходимо сделать коммит, который удаляет изменения, сохраненные нежелательным коммитом (рис. 32).

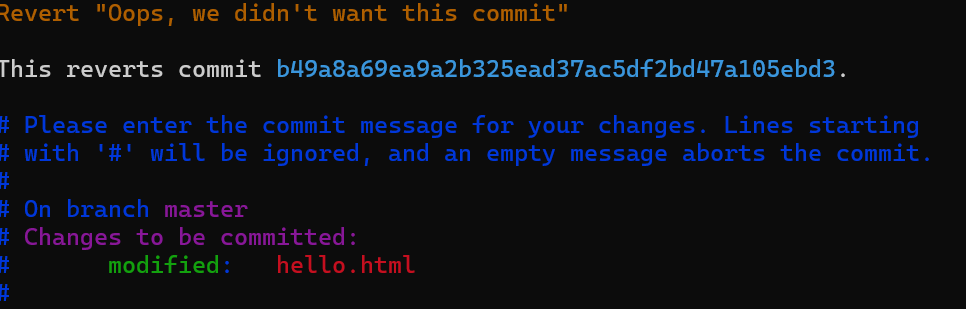


Рис. 32: Создание коммита с новыми изменениями, отменяющими предыдущие

Проверка лога показывает нежелательные и отмененные коммиты в наш репозиторий (рис. 33).



Рис. 33: Проверка лога

Сделаем быструю проверку нашей истории коммитов (рис. 34).

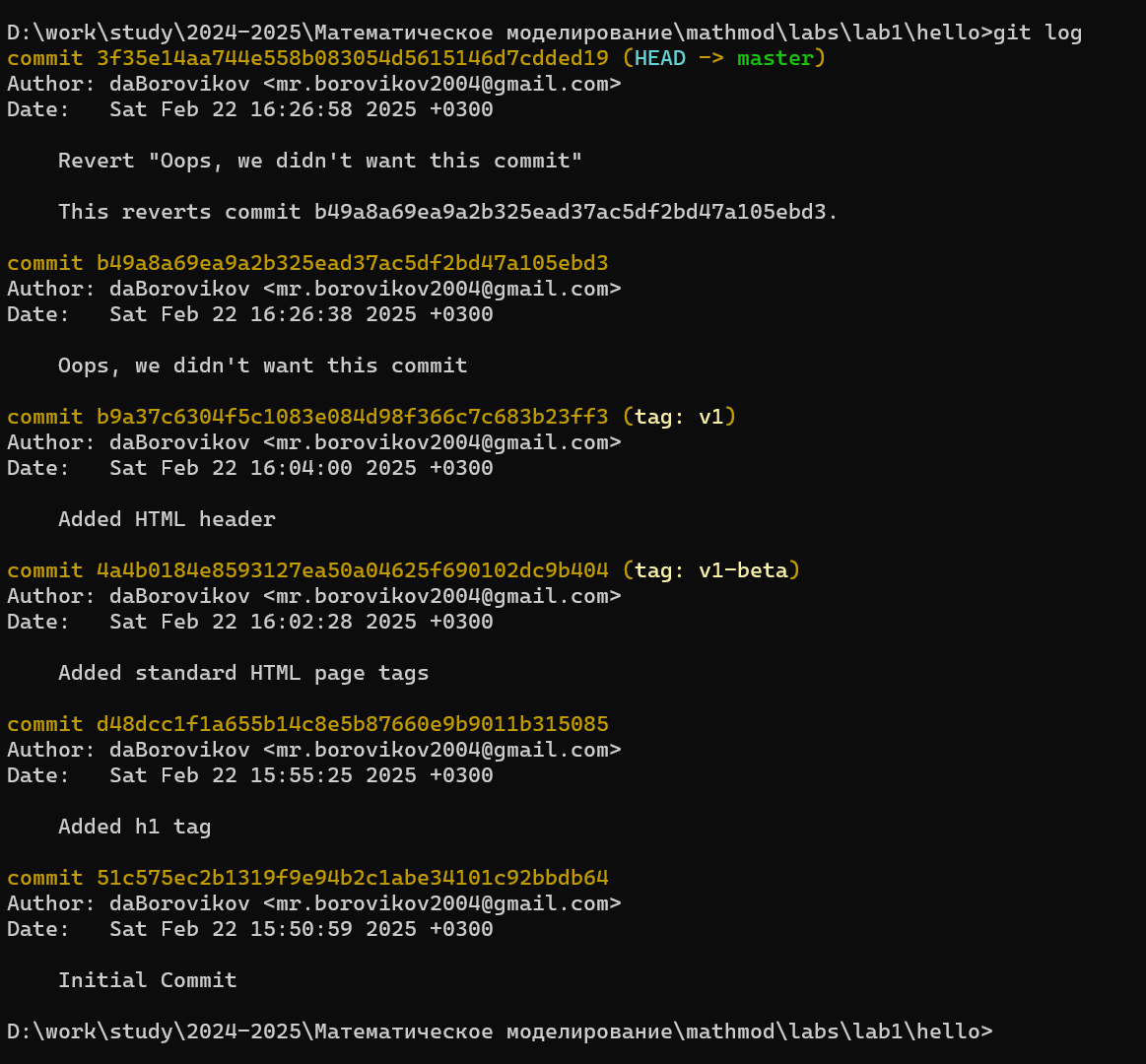


Рис. 34: Проверка нашей истории

Но прежде чем удалить коммиты, давайте отметим последний коммит тегом, чтобы потом можно было его найти (рис. 35).

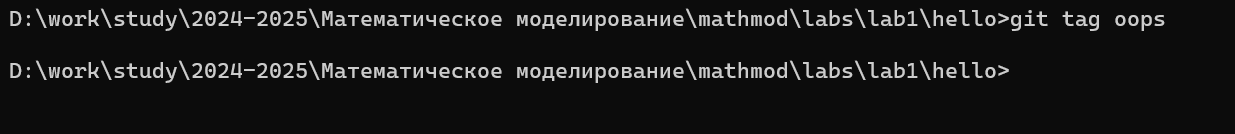


Рис. 35: Отметка этой ветки для начала

Глядя на историю лога, мы видим, что коммит с тегом «v1» является коммитом, предшествующим ошибочному коммиту. Давайте сбросим ветку до этой точки. Поскольку ветка имеет тег, мы можем использовать имя тега в команде сброса (если она не имеет тега, мы можем использовать хэш-значение) (рис. 36).

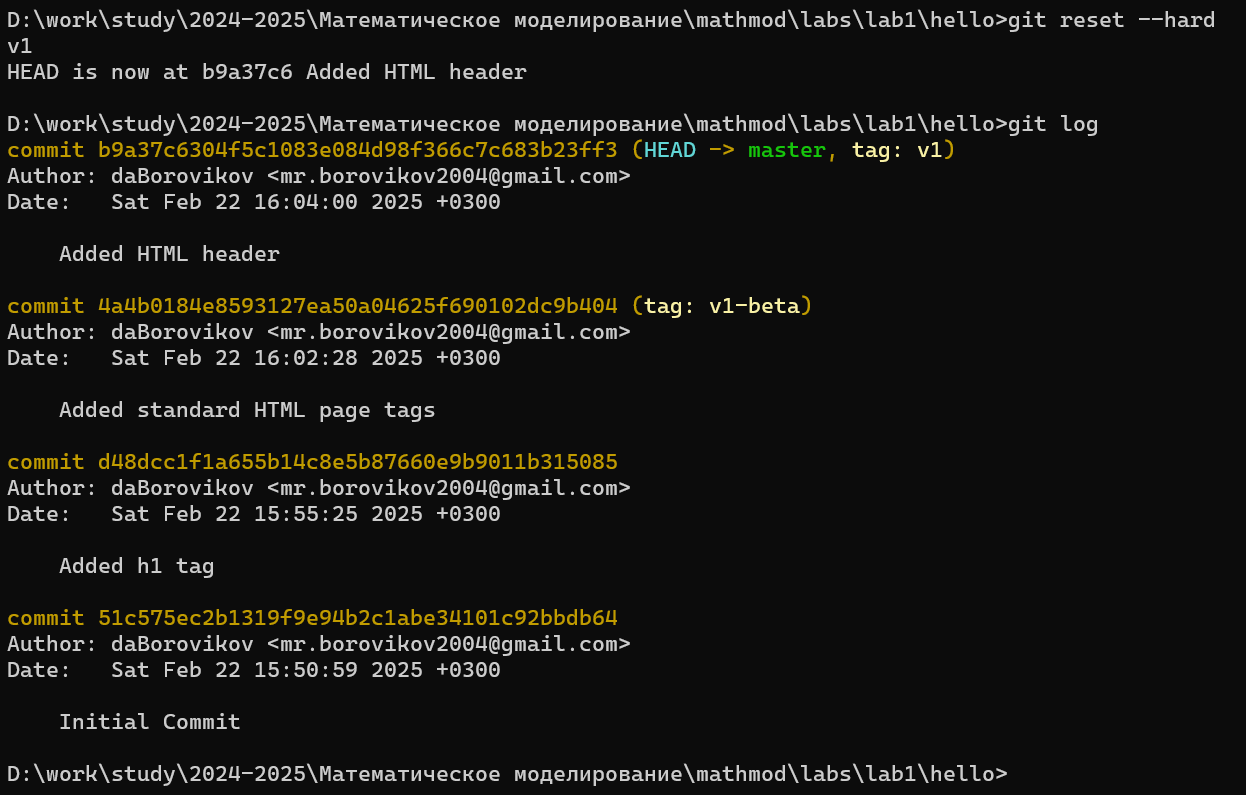


Рис. 36: Сброс коммитов к предшествующим коммиту Oops

Что же случается с ошибочными коммитами? Оказывается, что коммиты все еще находятся в репозитории. На самом деле, мы все еще можем на них ссылаться. Помните, в начале этого урока мы создали для отмененного коммита тег «oops». Давайте посмотрим на все коммиты (рис. 37).

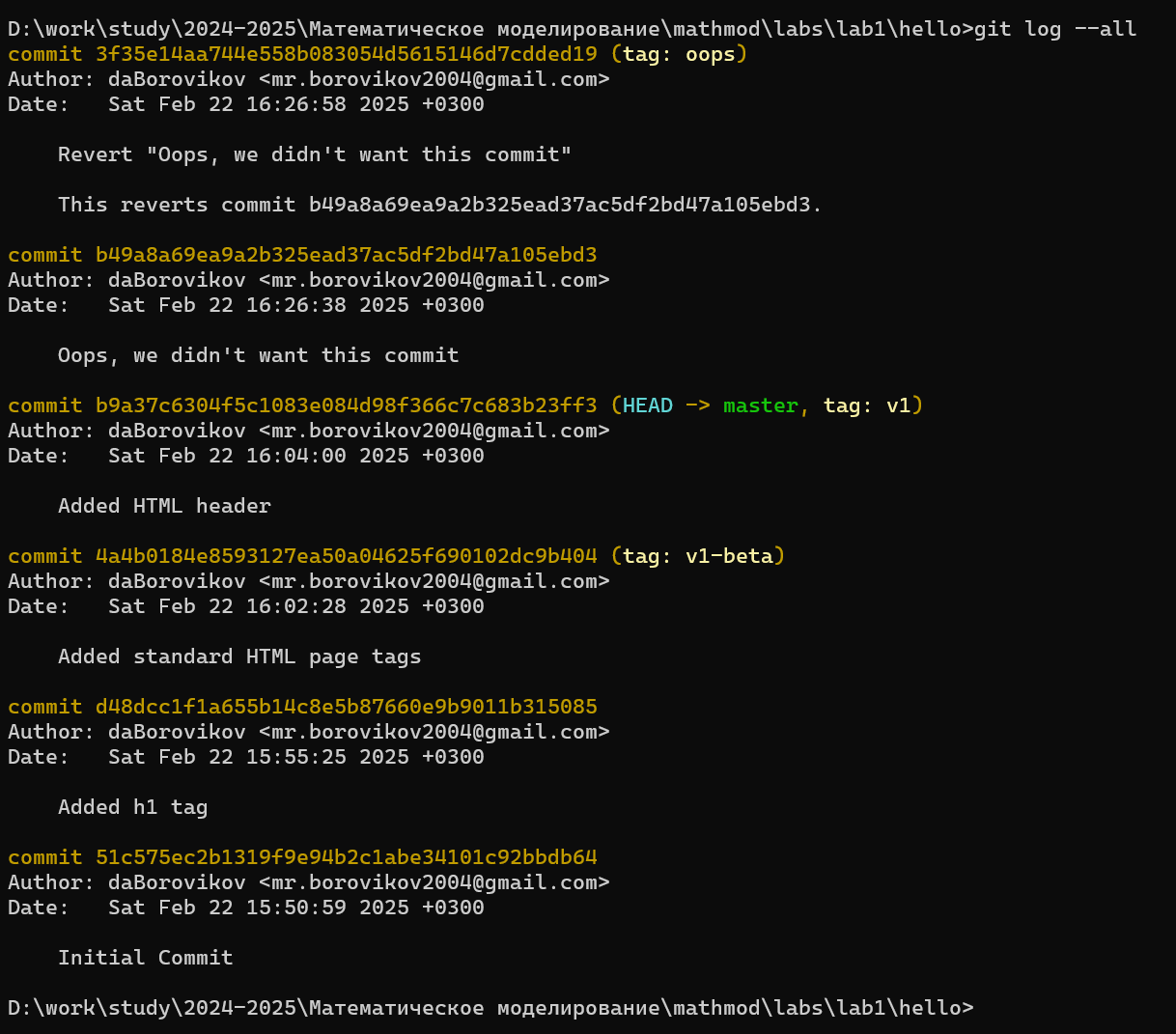


Рис. 37: Просмотр всех коммитов

Тег oops свою функцию выполнил. Давайте удалим его и коммиты, на которые он ссылался, сборщиком мусора (рис. 38).

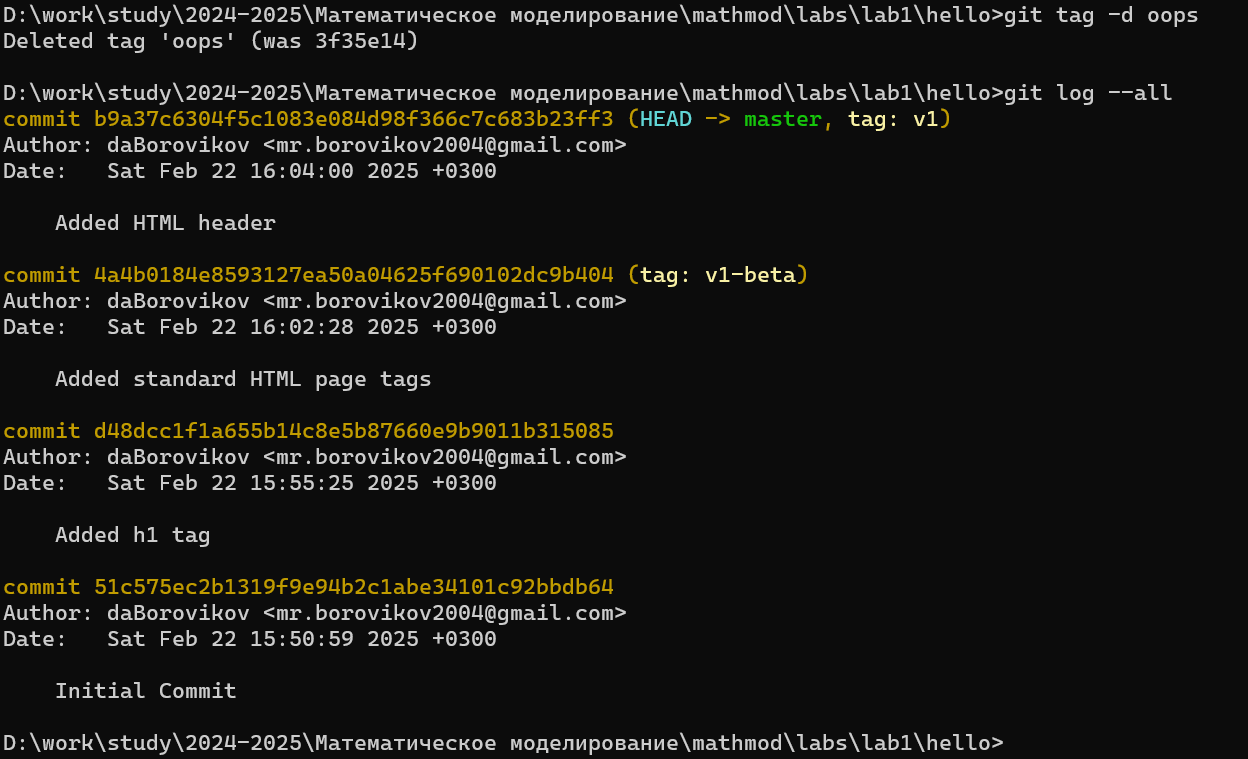


Рис. 38: Удаление тега oops

Добавим в страницу комментарий автора (вставим свою фамилию) (рис. 39).



Рис. 39: Добавление в страницу комментария автора

После чего выполним коммит (рис. 40).

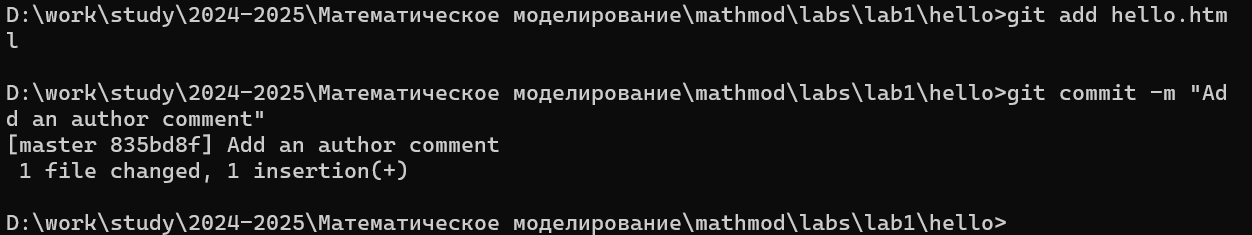


Рис. 40: Коммит

После совершения коммита мы понимаем, что любой хороший комментарий должен включать электронную почту автора. Обновим страницу hello, включив в нее email (рис. 41).

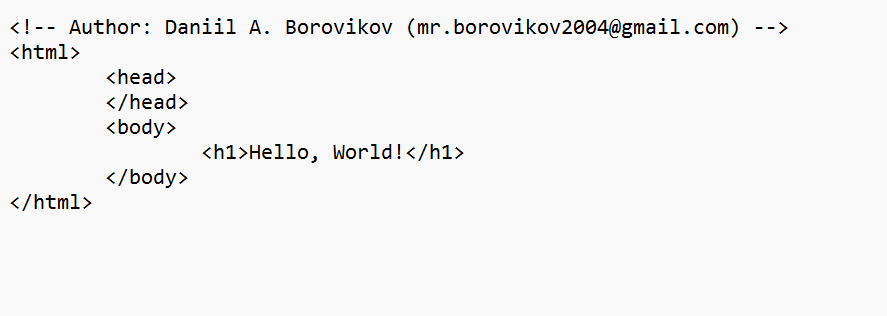


Рис. 41: Добавление email

Мы действительно не хотим создавать отдельный коммит только ради электронной почты. Давайте изменим предыдущий коммит, включив в него адрес электронной почты (рис. 42).

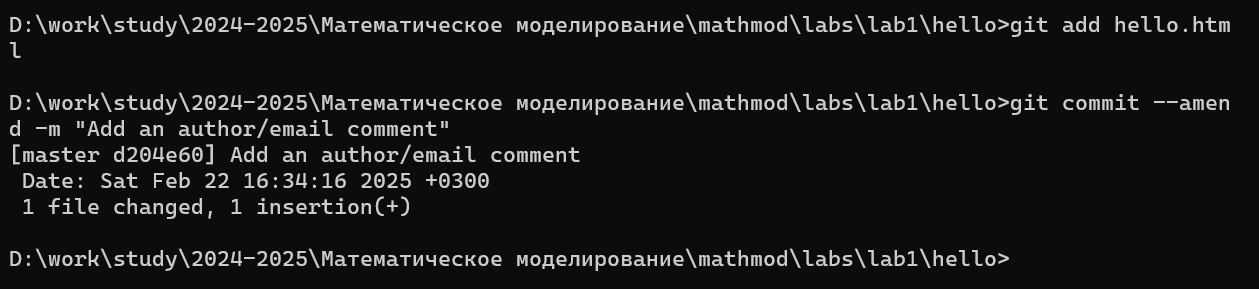


Рис. 42: Изменение предыдущего коммита

Просмотрим историю (рис. 43).

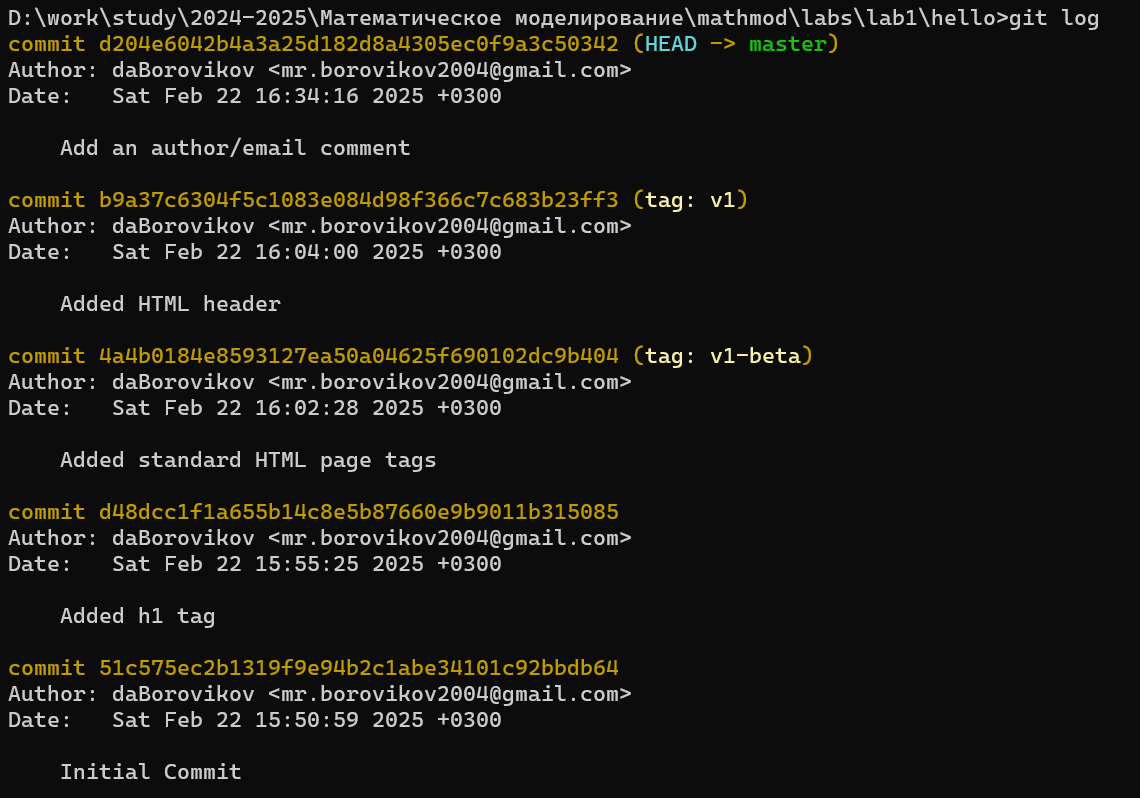


Рис. 43: Просмотр истории

Сейчас мы собираемся создать структуру нашего репозитория. Давайте перенесем страницу в каталог lib (рис. 44).

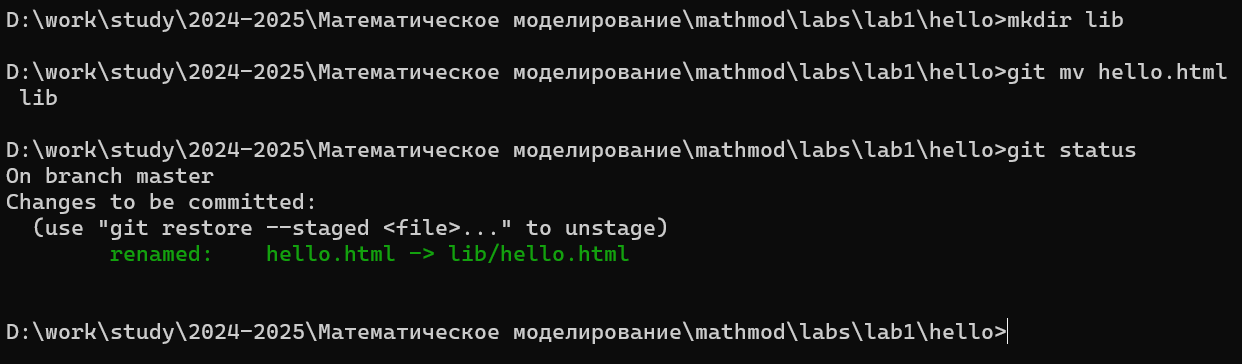


Рис. 44: Перемещение файла hello.html в каталог lib

Давайте сделаем коммит этого перемещения (рис. 45).

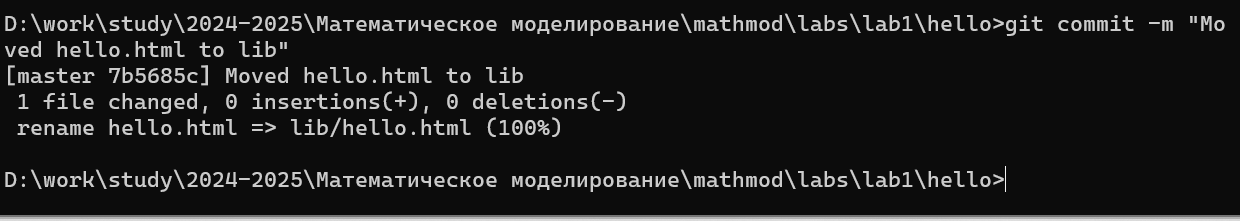


Рис. 45: Коммит в новом каталоге

Добавим файл index.html в наш репозиторий, добавим содержимое в файл и сделаем коммит (рис. 46).

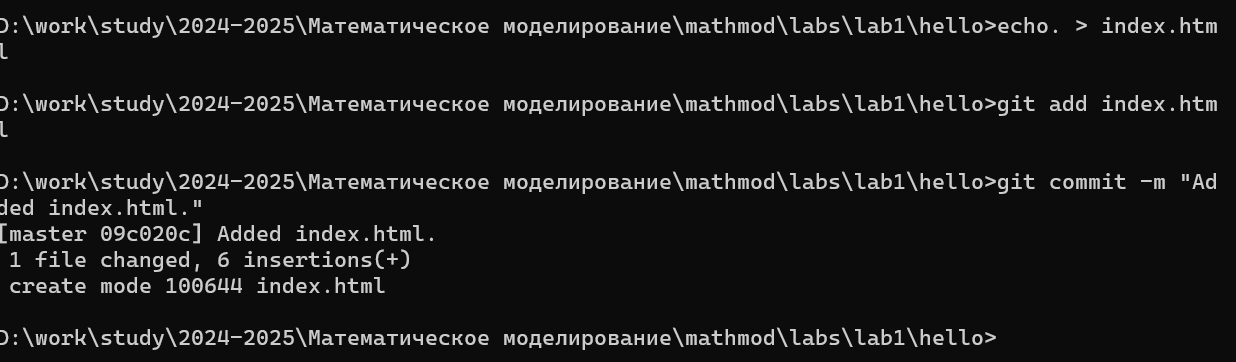


Рис. 46: Добавление index.html

Просмотрим каталог, в котором хранится вся информация git (рис. 47).

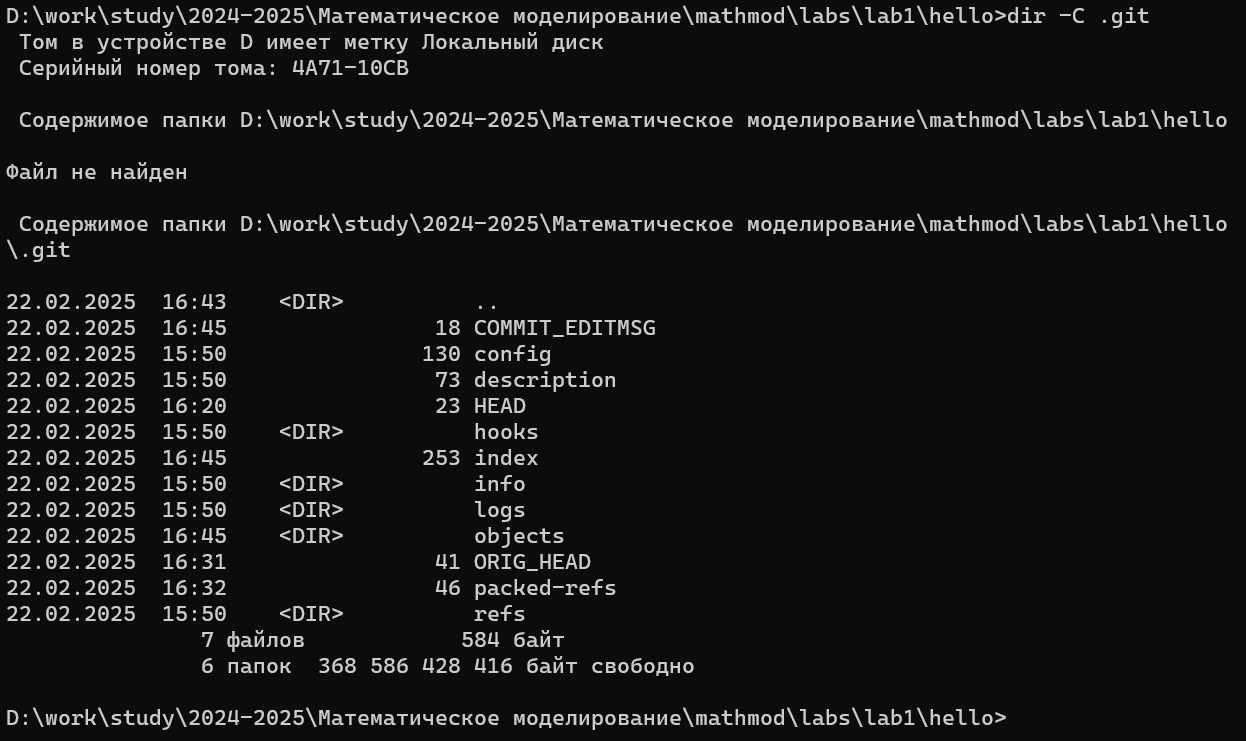


Рис. 47: Каталог .git

Далее просмотрим набор каталогов, имена которых состоят из 2 символов. Имена каталогов являются первыми двумя буквами хэша sha1 объекта, хранящегося в git (рис. 48).

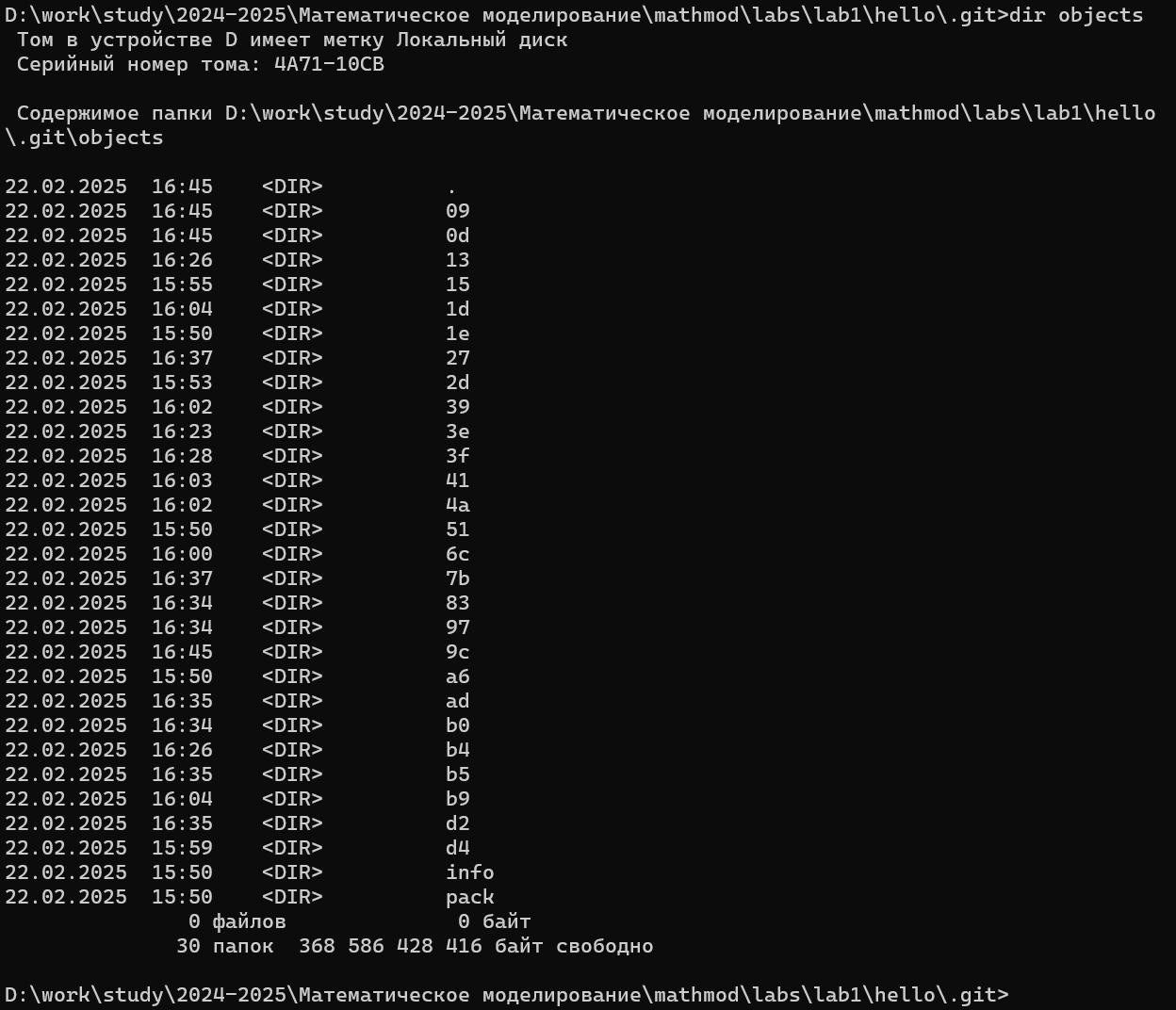


Рис. 48: Просмотр базы данных объектов

Осмотрим один из каталогов с именем из 2 букв. Мы видим файлы с именами из 38 символов. Это файлы, содержащие объекты, хранящиеся в git. Они сжаты и закодированы, поэтому просмотр их содержимого нам мало чем поможет (рис. 49).

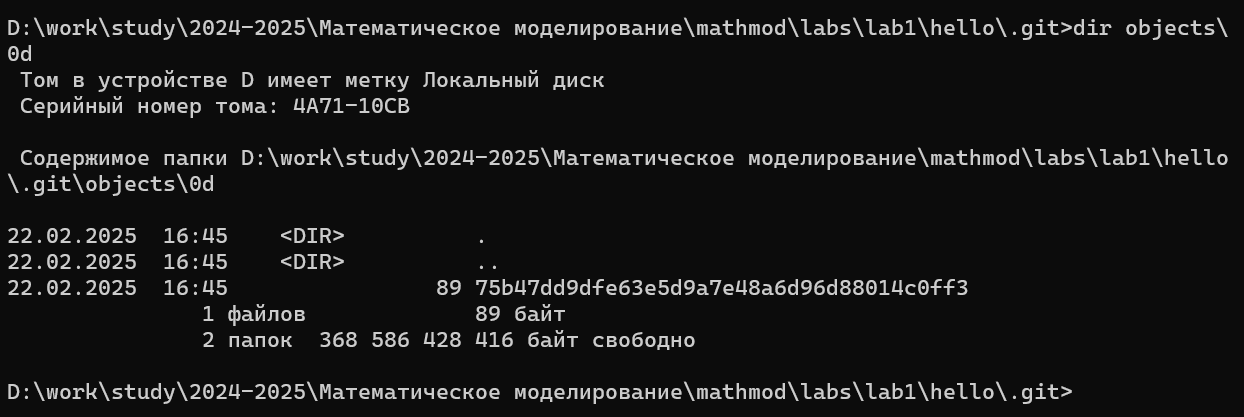


Рис. 49: Углубление в базу данных объектов

Изучим файл конфигурации, создающийся для каждого конкретного проекта. Записи в этом файле будут перезаписывать записи в файле .gitconfig нашего главного каталога, по крайней мере в рамках этого проекта (рис. 50).

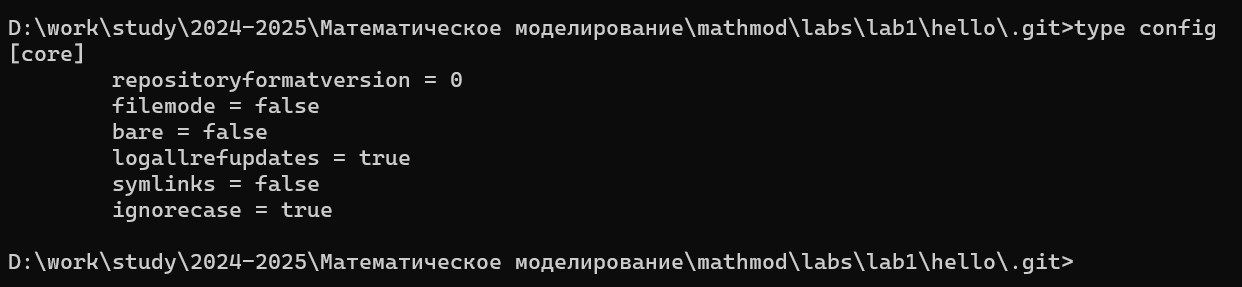


Рис. 50: Просмотр Config File

Выполним набор команд для папки .git (рис. 51).

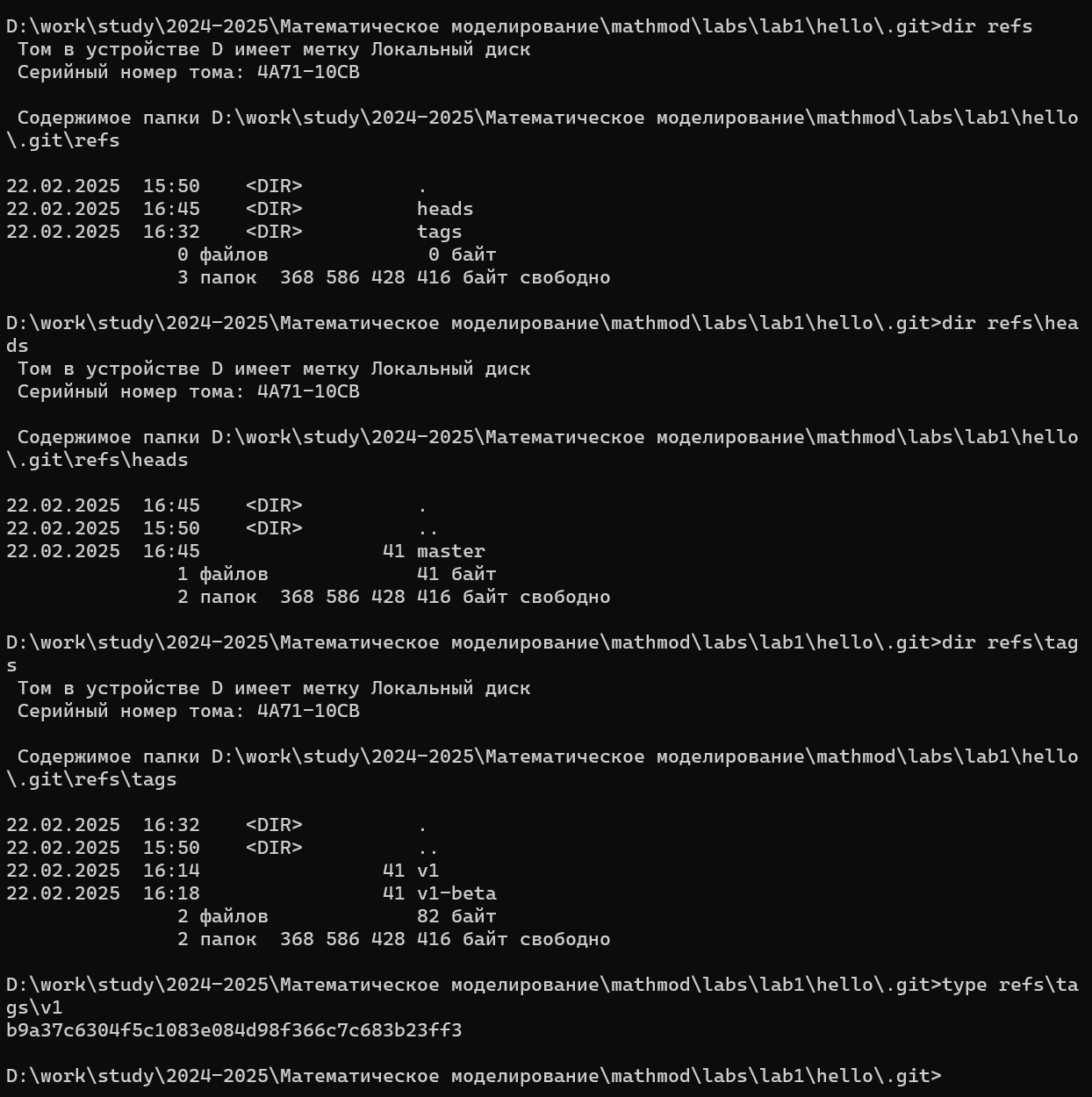


Рис. 51: Изучение веток и тегов

Файл HEAD содержит ссылку на текущую ветку, в данный момент это должна быть ветка master (рис. 52).

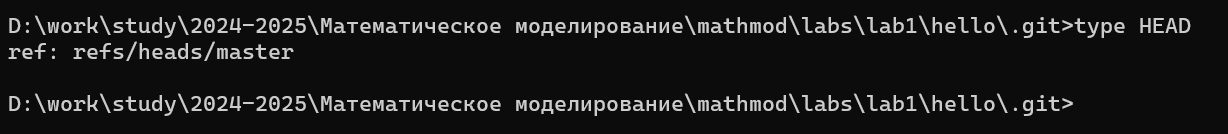


Рис. 52: Файл HEAD

Эта команда показывает последний коммит в репозиторий (рис. 53).

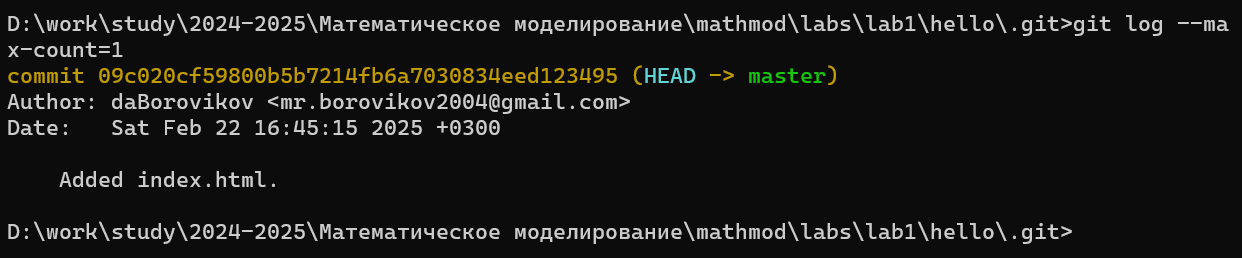


Рис. 53: Поиск последнего коммита

Выведем последний коммит с помощью SHA1 хэша (рис. 54).

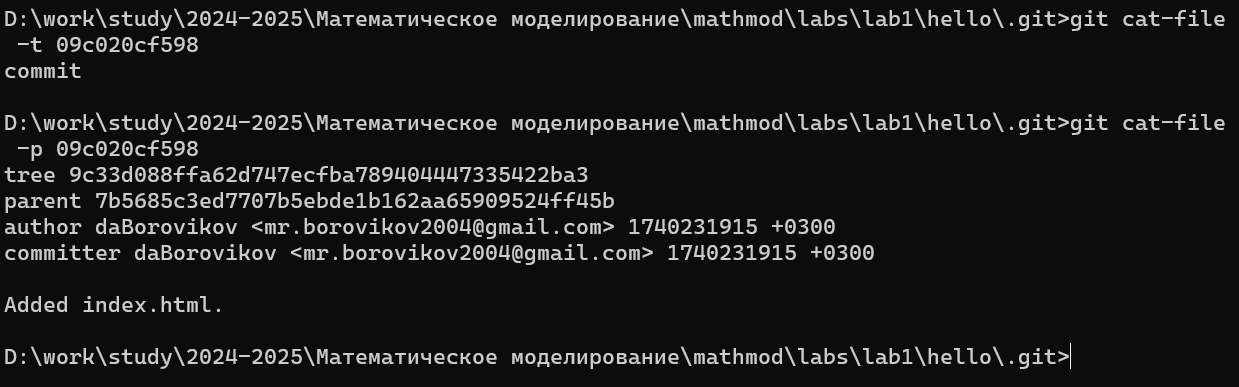


Рис. 54: Вывод последнего коммита с помощью SHA1 хэша

Мы можем вывести дерево каталогов, ссылка на который идет в коммите. Это должно быть описание файлов (верхнего уровня) в нашем проекте (для конкретного коммита). Используем SHA1 хэш из строки «дерева», из списка выше (рис. 55).

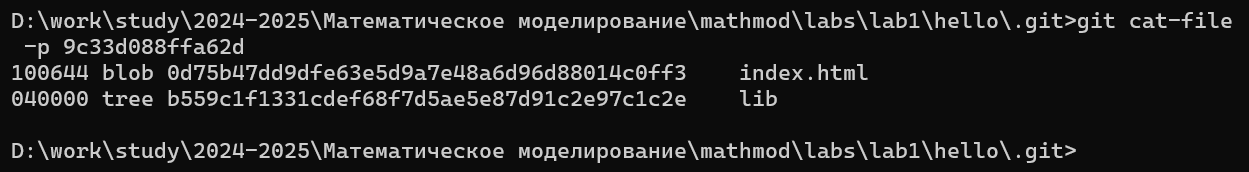


Рис. 55: Поиск дерева

Выведем каталог lib (рис. 56).

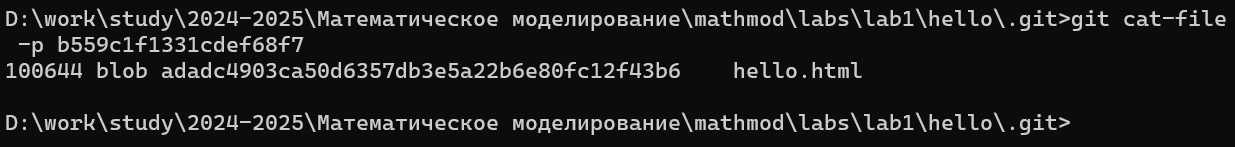


Рис. 56: Вывод каталога lib

Выведем файл hello.html (рис. 57).

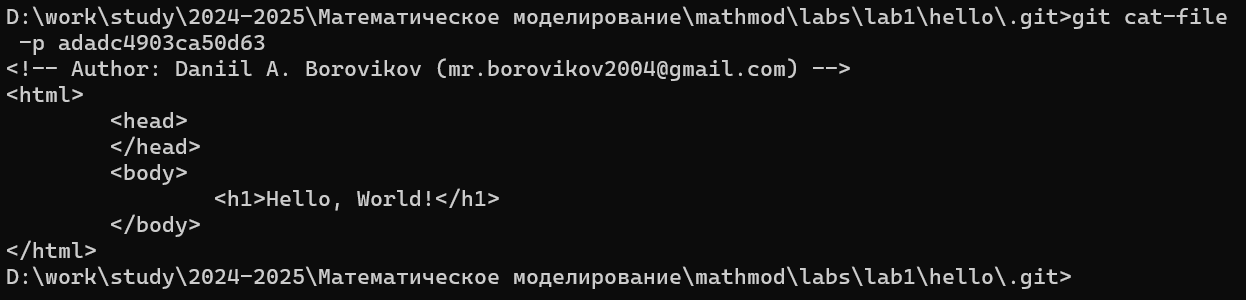


Рис. 57: Вывод файла hello.html

Давайте назовем нашу новую ветку «style» (рис. 58).

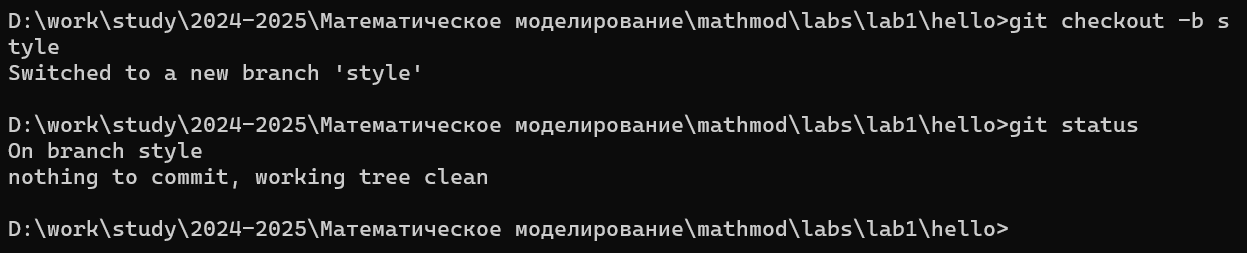


Рис. 58: Создание ветки

Создадим файл css (рис. 59).

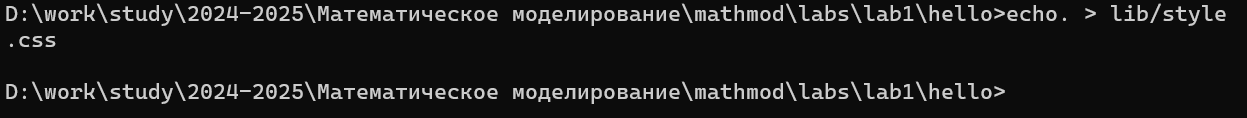


Рис. 59: Создание файла css

Добавим содержимое в файл (рис. 60).

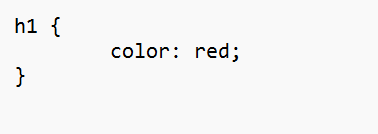


Рис. 60: Добавление содержимого в файл

Выполним коммит (рис. 61).

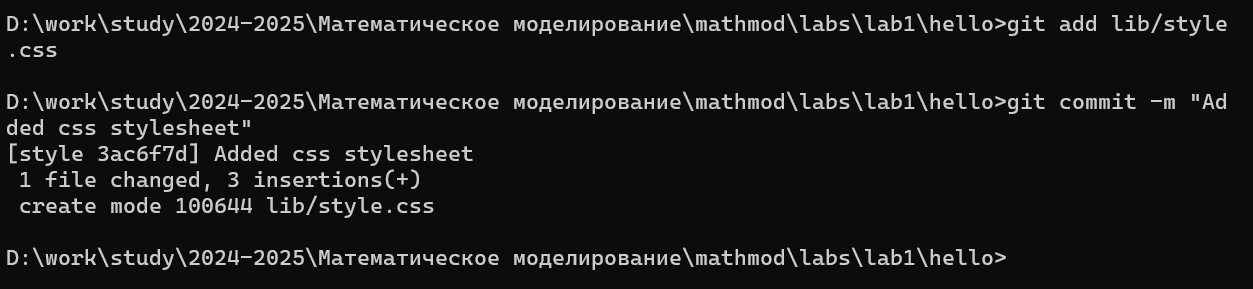


Рис. 61: Коммит

Обновим файл hello.html, чтобы использовать стили style.css (рис. 62).

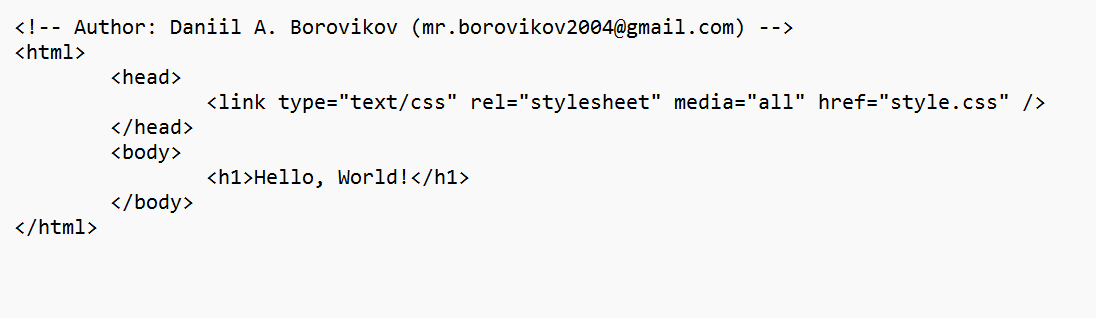


Рис. 62: Обновление файла hello.html

Выполним коммит (рис. 63).

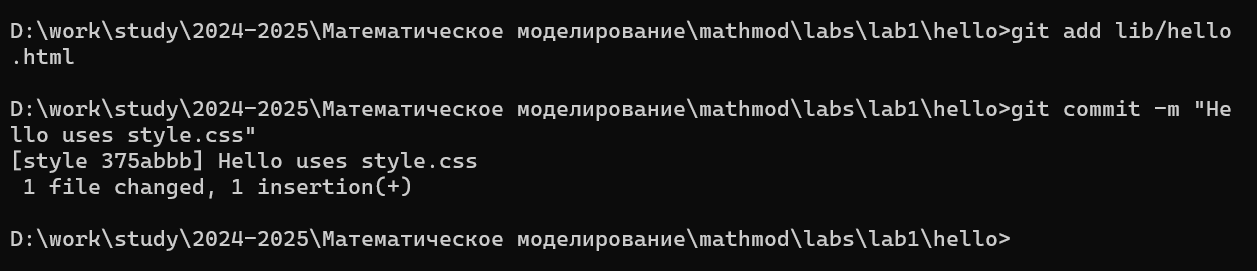


Рис. 63: Коммит

Обновим файл index.html, чтобы он тоже использовал style.css (рис. 64).

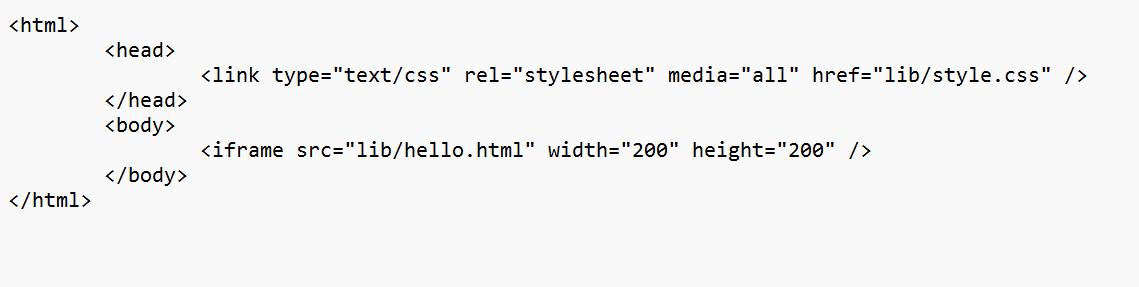


Рис. 64: Обновление файла index.html

Выполним коммит (рис. 65).

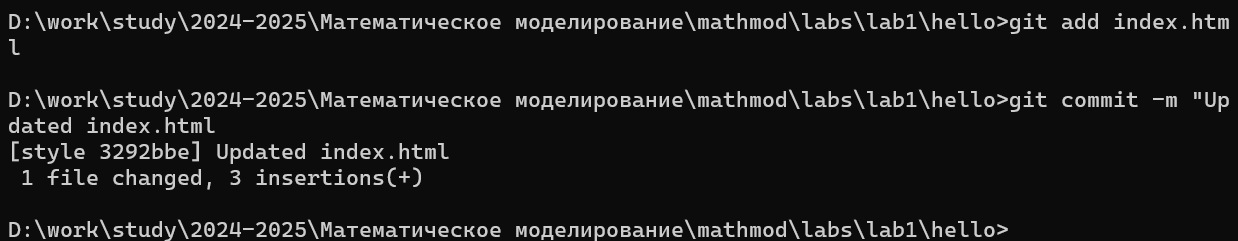


Рис. 65: Коммит

Теперь в нашем проекте есть две ветки (рис. 66).



Рис. 66: Навигация по веткам

Используем команду git checkout для переключения между ветками (рис. 67).

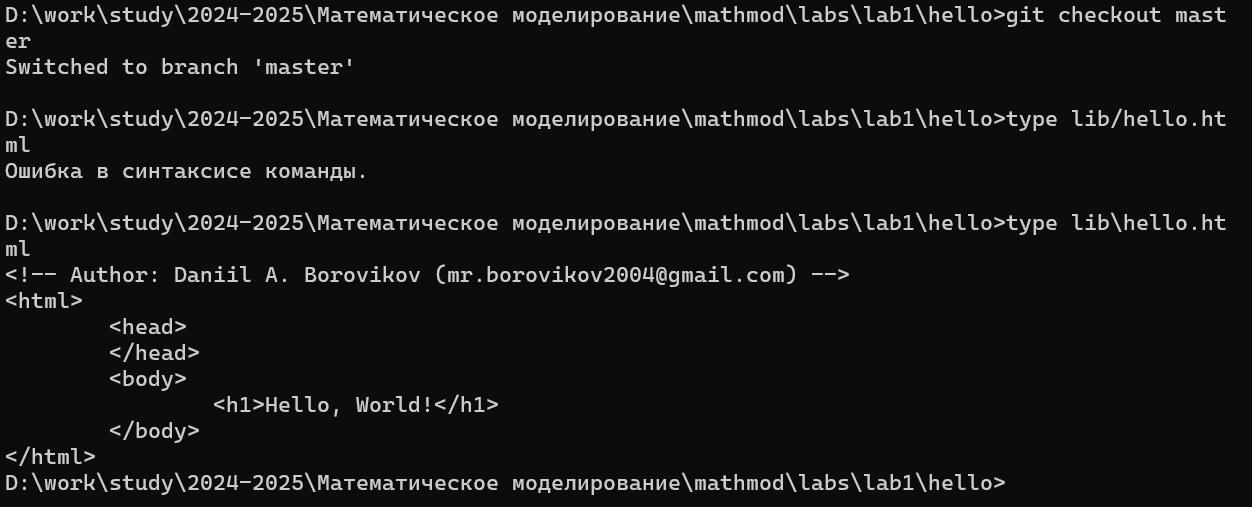


Рис. 67: Переключение на ветку master

Вернемся к ветке style (рис. 68).

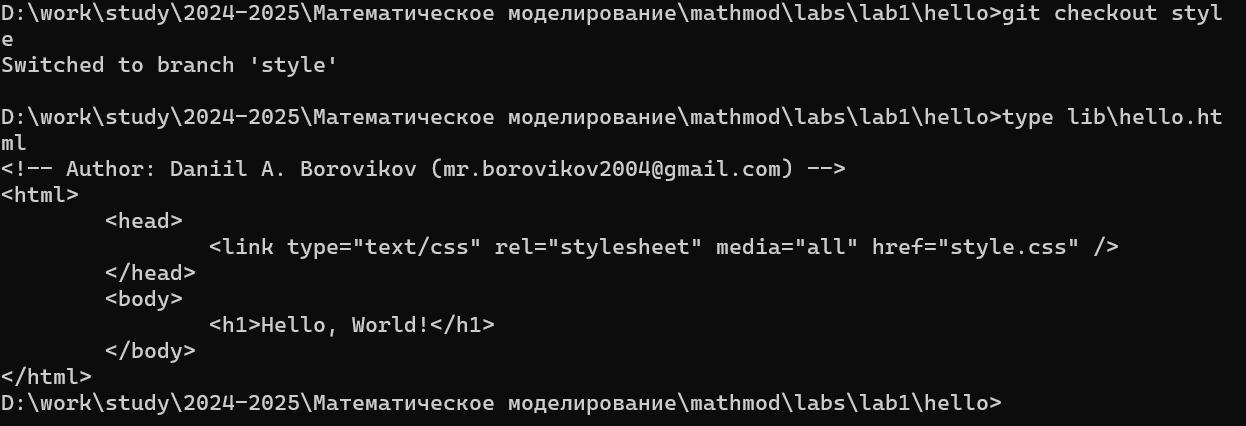


Рис. 68: Переключение на ветку style

Переключимся на ветку master и создадим файл README.md (рис. 69).

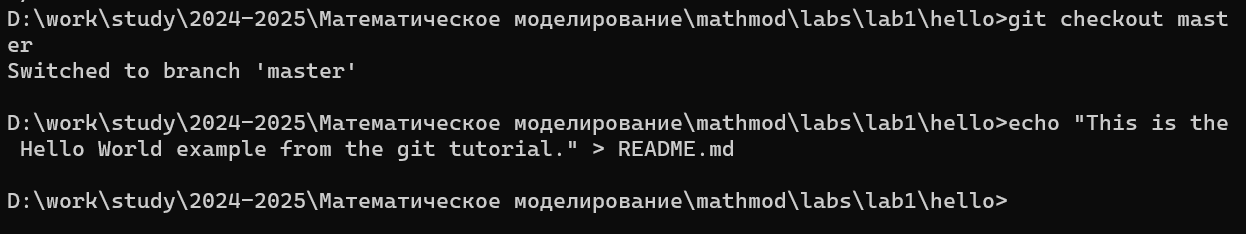


Рис. 69: Создание файла README.md

Сделаем коммит изменений README.md в ветку master (рис. 70).

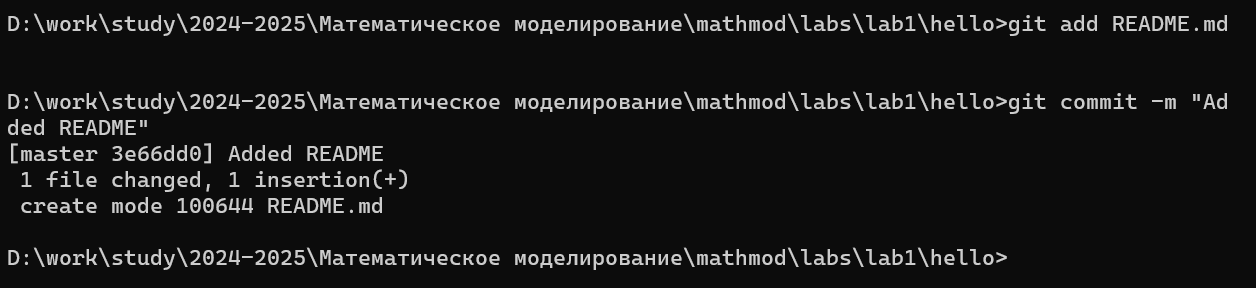


Рис. 70: Коммит изменений README.md в ветку master

Просмотрим текущие ветки (рис. 71).

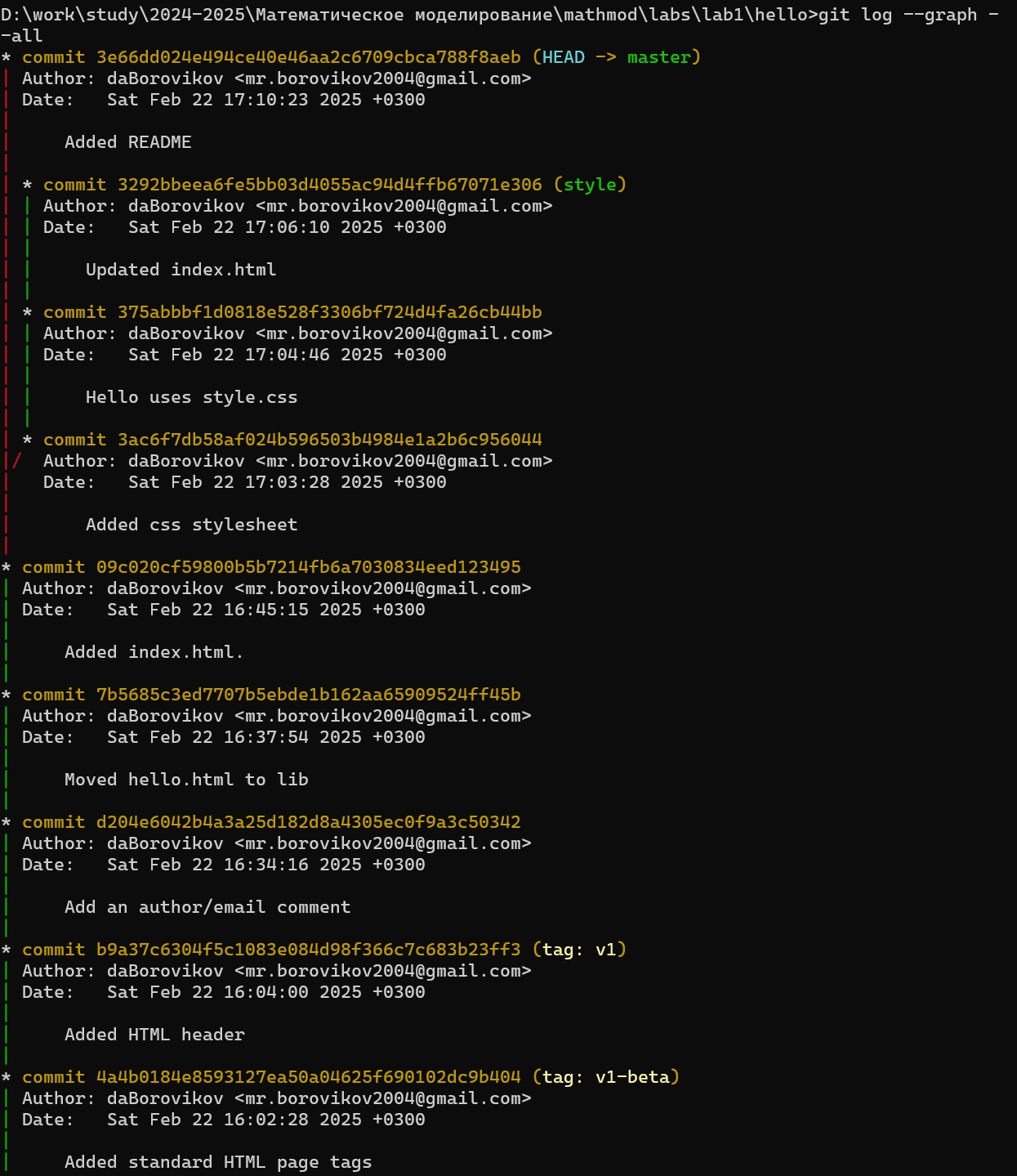


Рис. 71: Просмотр текущих веток

Слияние переносит изменения из двух веток в одну. Давайте вернемся к веткет style и сольем master с style (рис. 72).

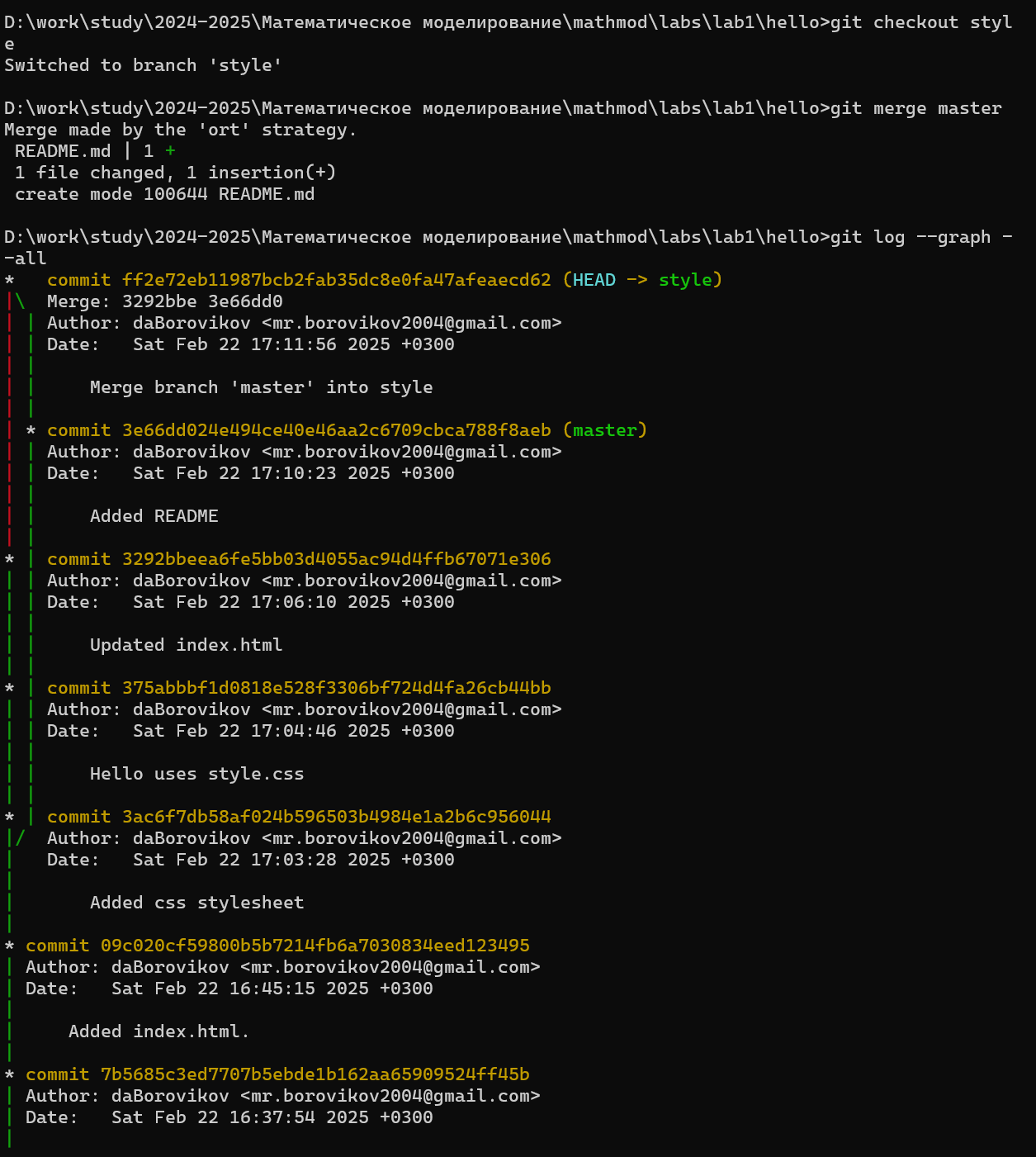


Рис. 72: Слияние веток

Вернёмся в ветку master (рис. 73).

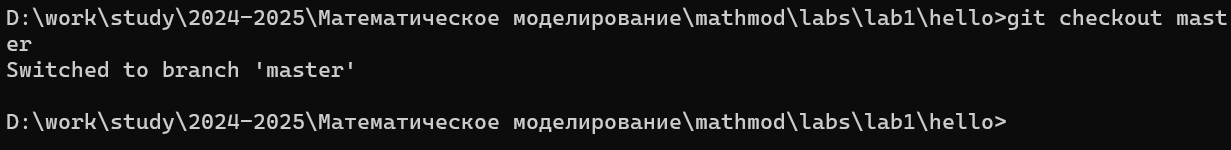


Рис. 73: Возвращение в ветку master

Внесём следующие изменения в файл (рис. 74).

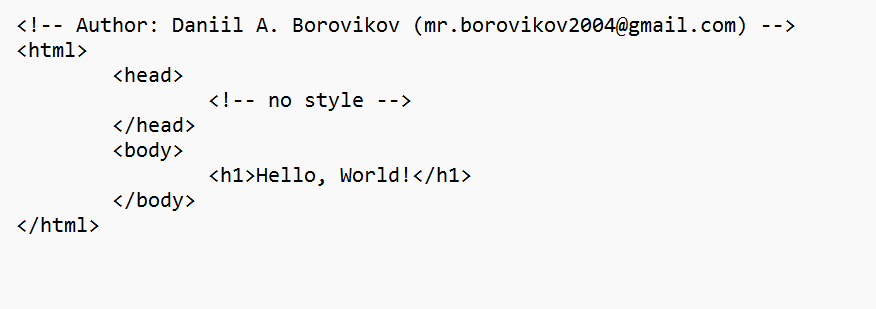


Рис. 74: Внесение изменений в файл

Выполним коммит (рис. 75).

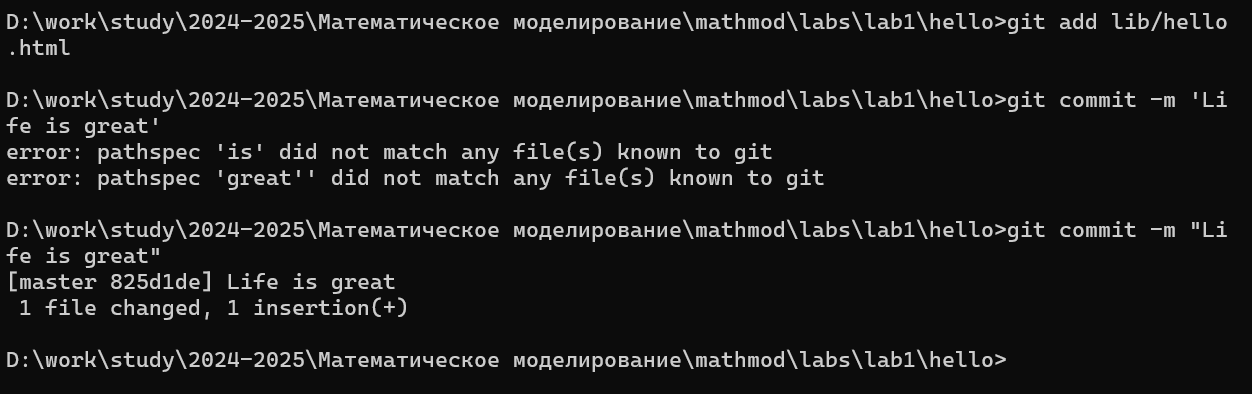


Рис. 75: Коммит

Просмотрим ветки (рис. 76).

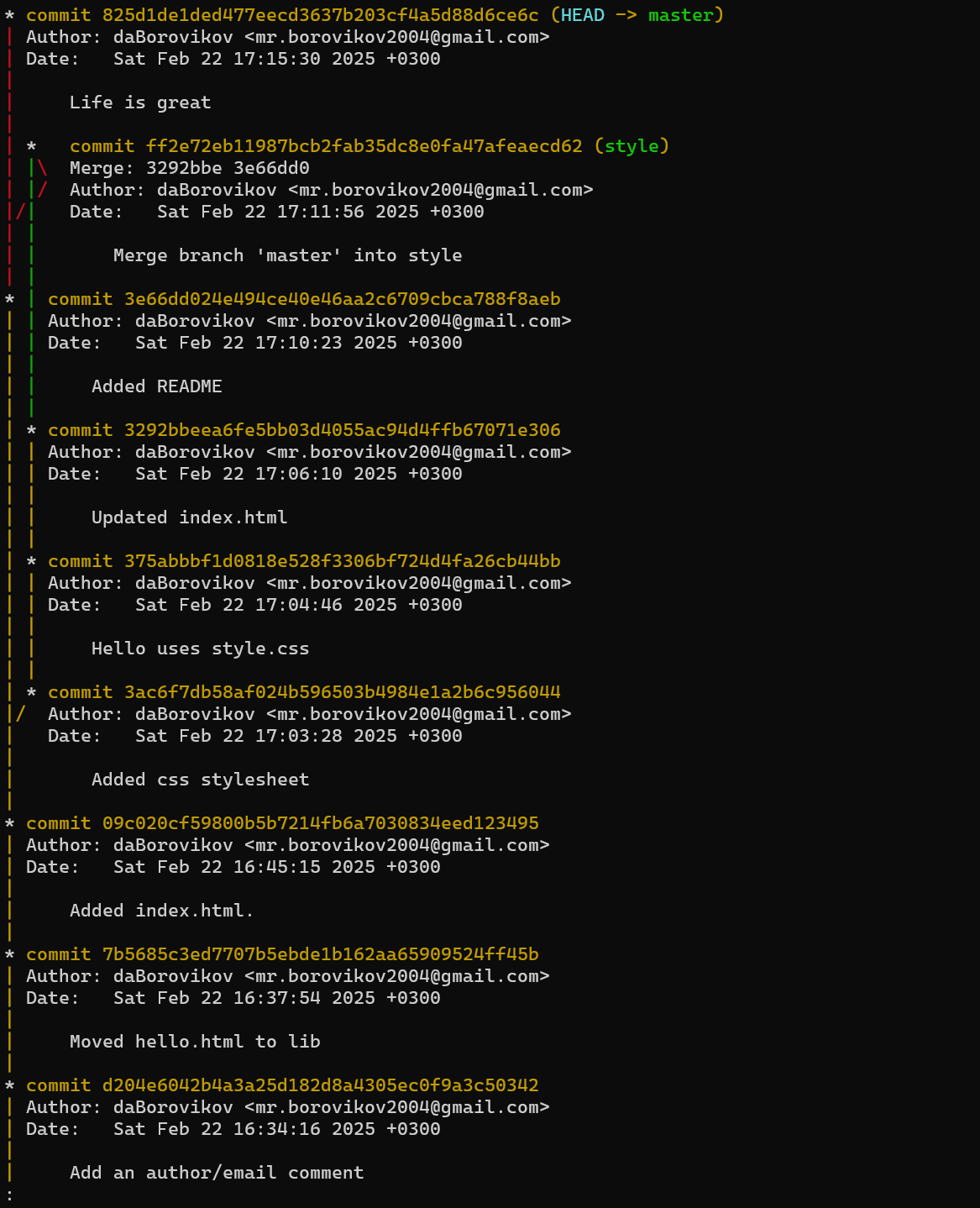


Рис. 76: Просмотр веток

Теперь вернемся к ветке style и попытаемся объединить ее с новой веткой master (рис. 77).

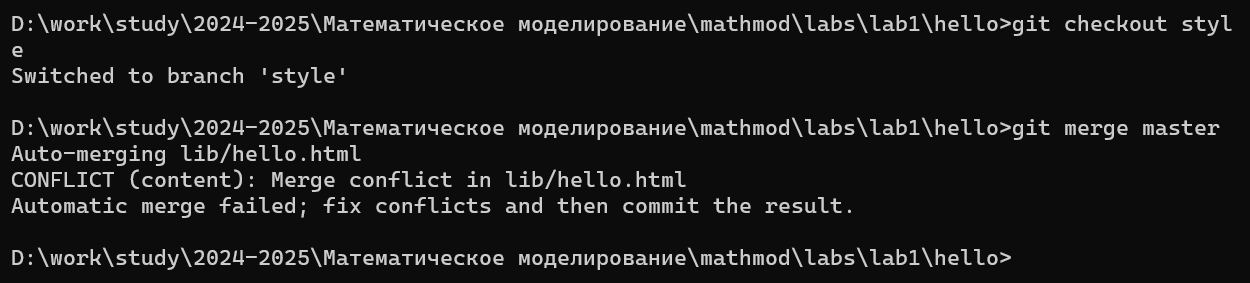


Рис. 77: Попытка объединения

Нам необходимо вручную разрешить конфликт. Внесём изменения в lib/hello.html для достижения следующего результата (рис. 78).

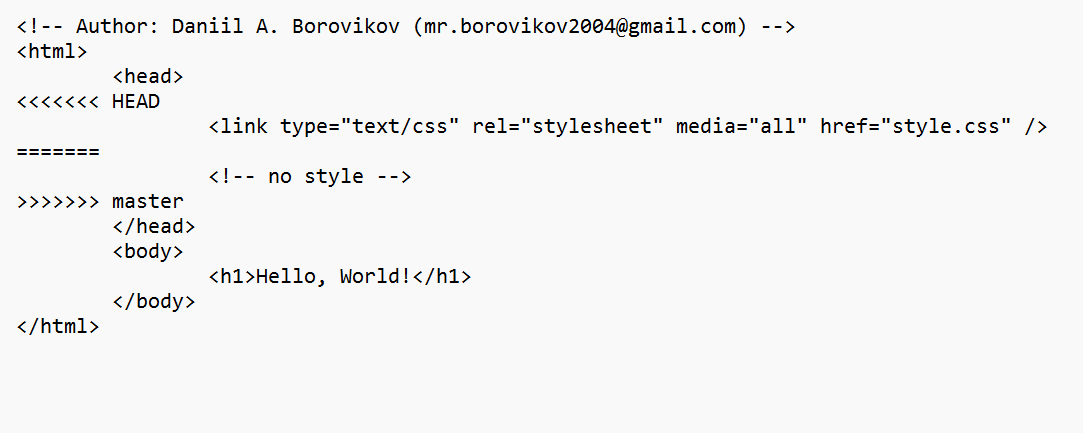


Рис. 78: Решение конфликта

Сделаем коммит решения конфликта (рис. 79).

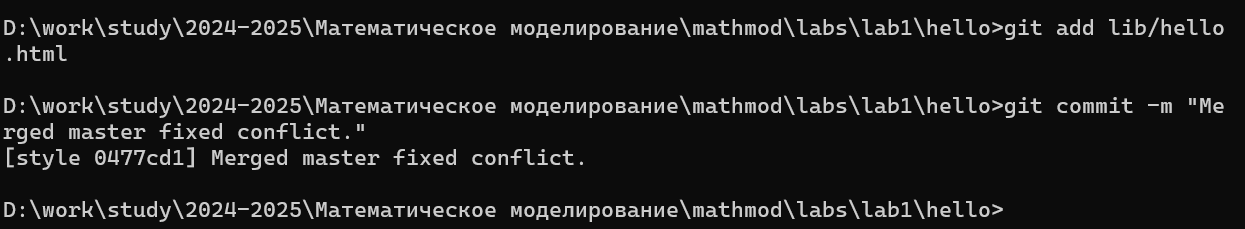


Рис. 79: Коммит решения конфликта

Вернемся на ветке style к точке перед тем, как мы слили ее с веткой master. Мы можем сбросить ветку к любому коммиту. По сути, это изменение указателя ветки на любую точку дерева коммитов. В этом случае мы хотим вернуться в ветке style в точку перед слиянием с master. Нам необходимо найти последний коммит перед слиянием (рис. 80).

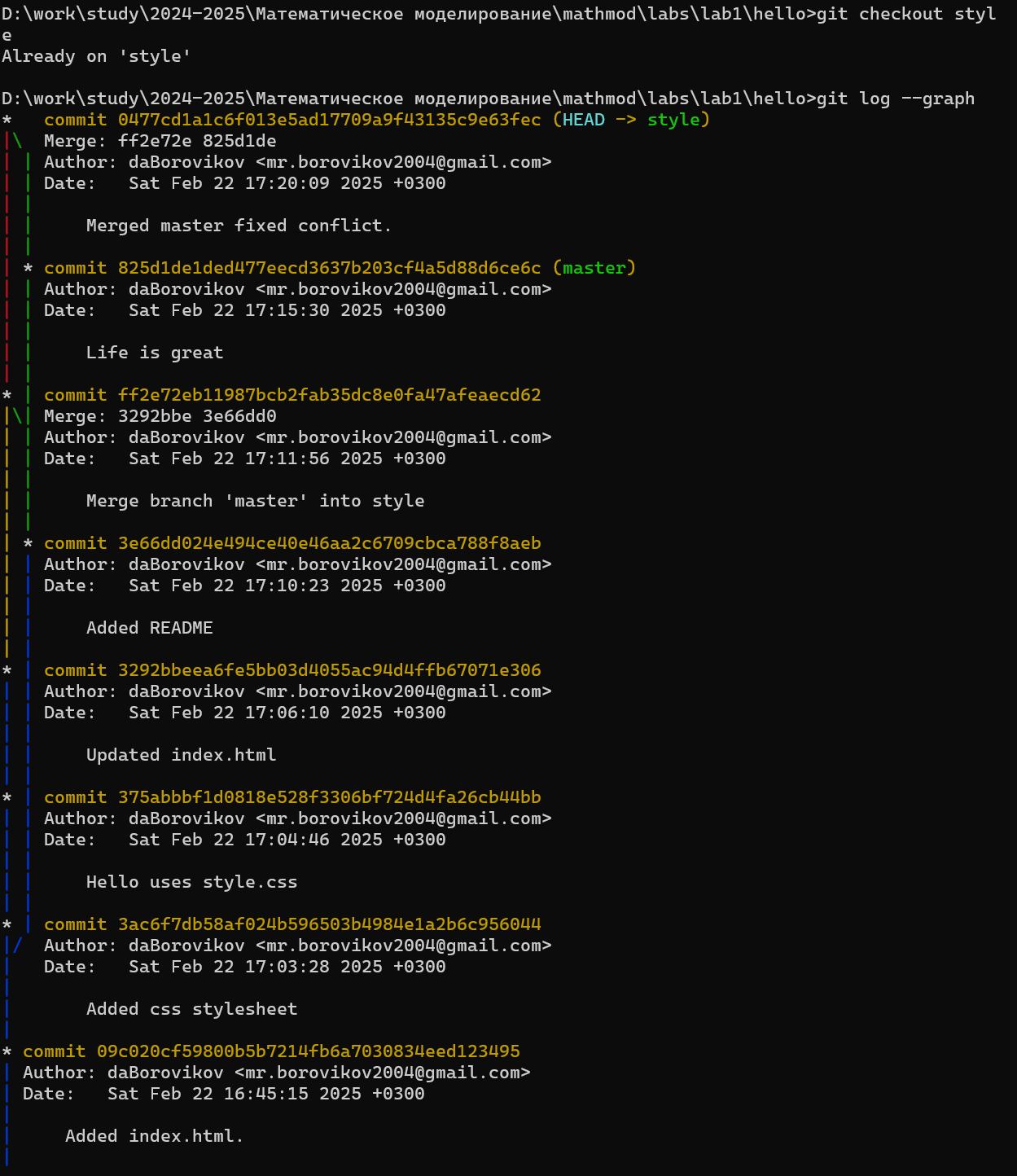


Рис. 80: Возвращение на ветку style

Сбросим ветку style к нужному коммиту (рис. 81).

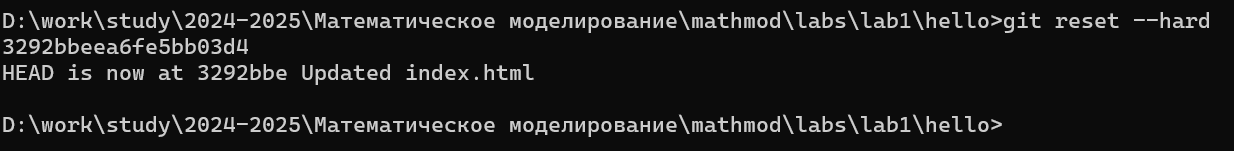


Рис. 81: Сброс ветки style к нужному коммиту

Проверим лог ветки style. У нас в истории больше нет коммитов слияний. (рис. 82).

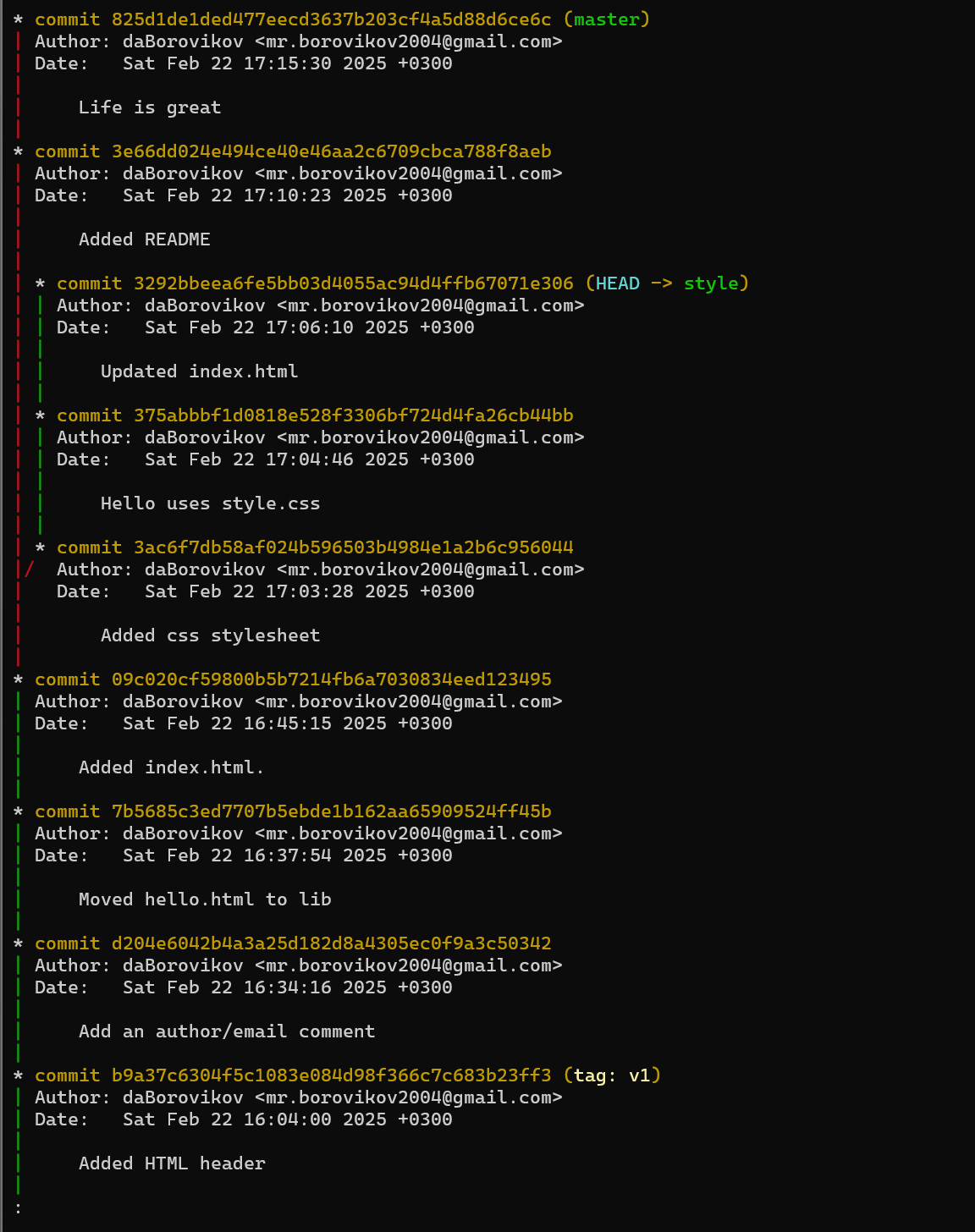


Рис. 82: Проверка ветки

Сбросим ветку master (рис. 83).



Рис. 83: Сброс ветки master

Используем команду rebase вместо команды merge. Мы вернулись в точку до первого слияния и хотим перенести изменения из ветки master в нашу ветку style. На этот раз для переноса изменений из ветки master мы будем использовать команду git rebase вместо слияния (рис. 84).

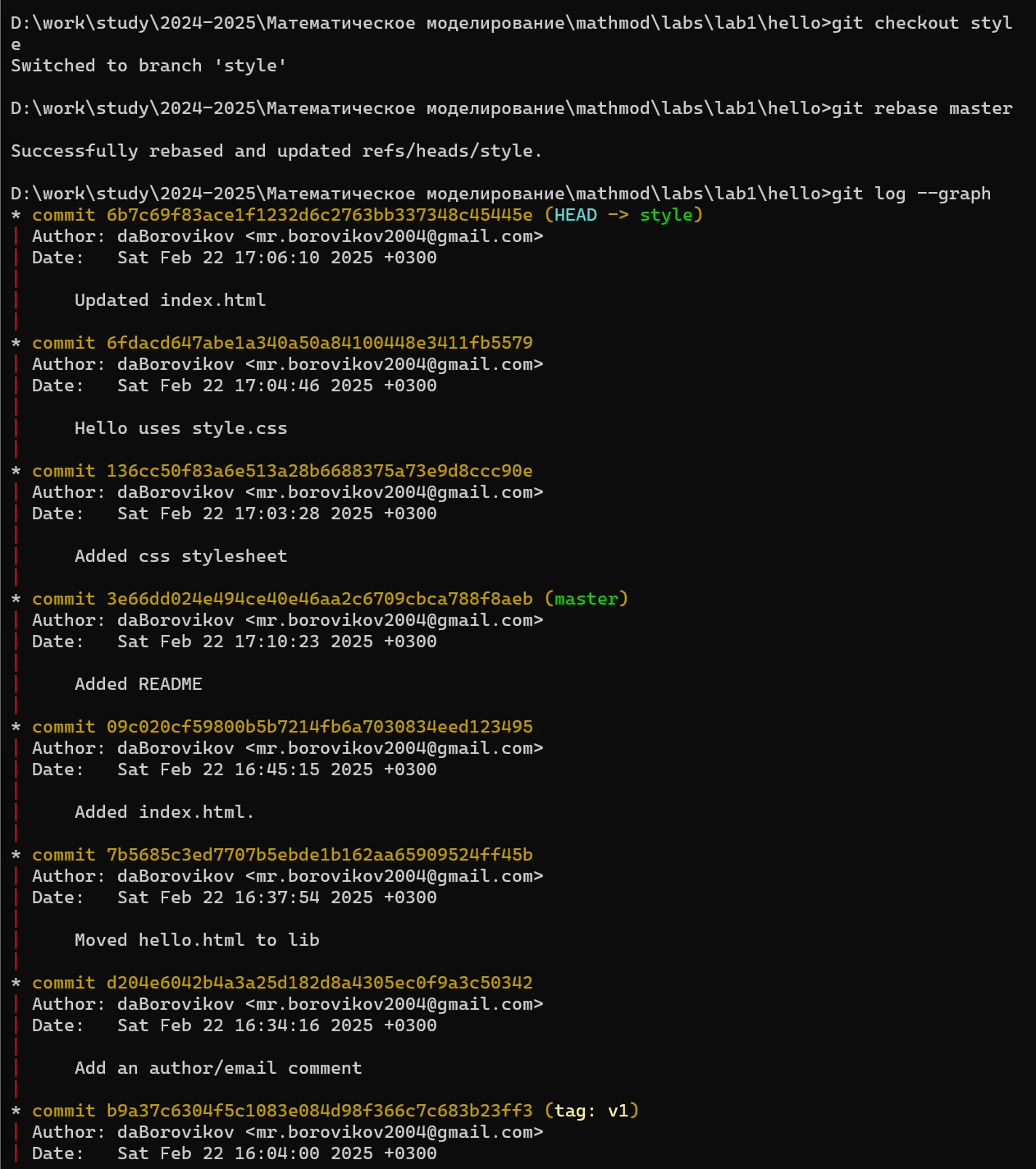


Рис. 84: Перебазирование

Выполним слияние style в master (рис. 85).

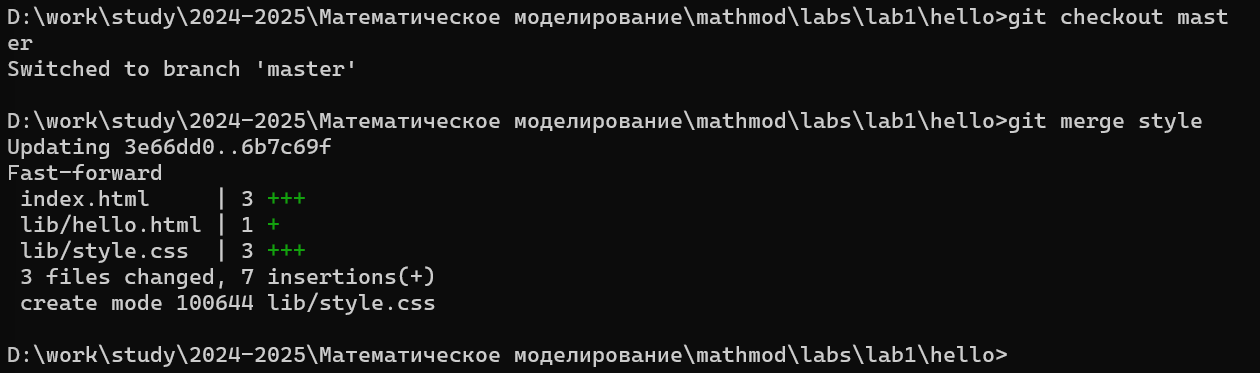


Рис. 85: Слияние style в master

Просмотрим логи (рис. 86).

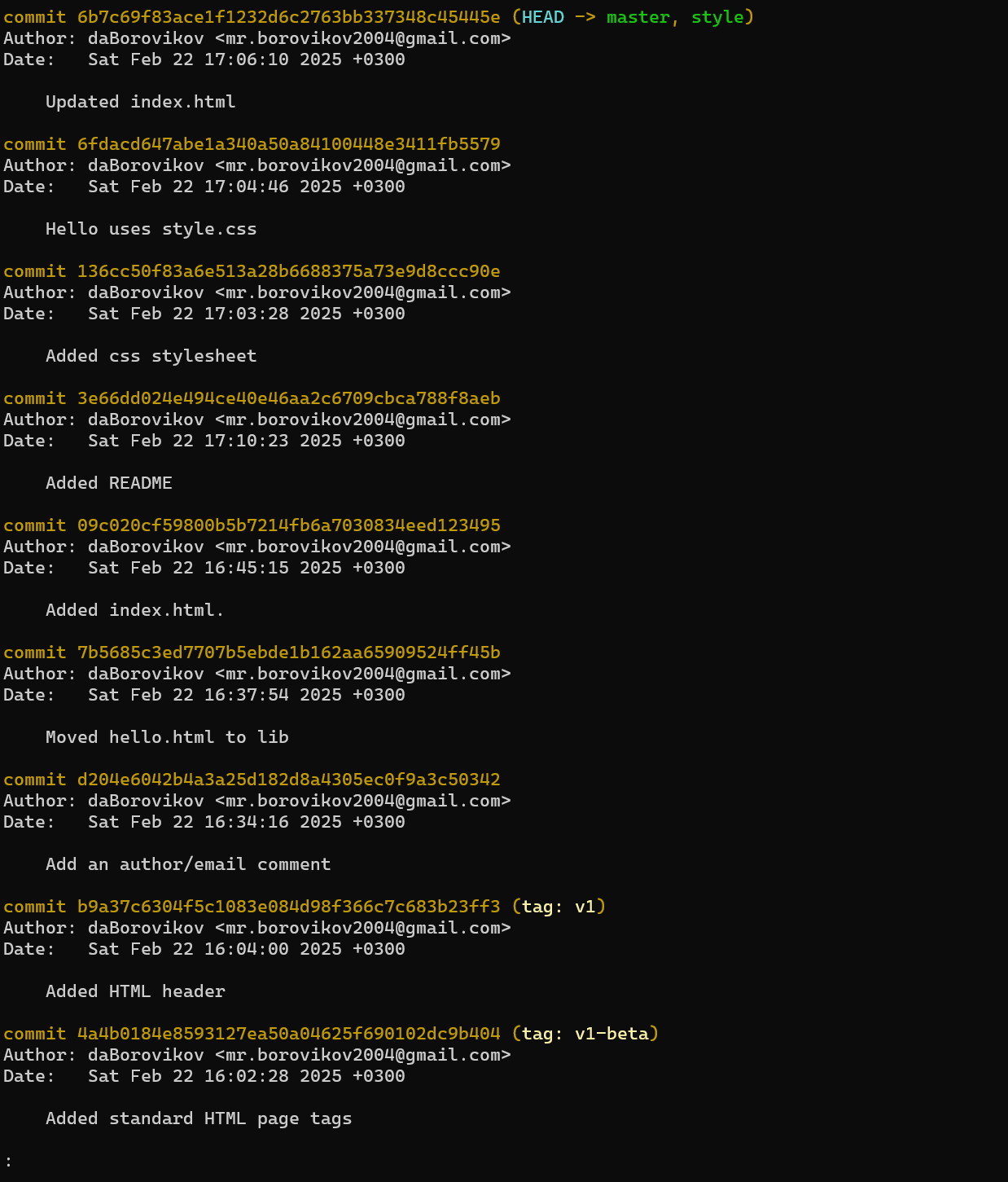


Рис. 86: Просмотр логов

Перейдём в рабочий каталог и сделаем клон нашего репозитория hello (рис. 87).

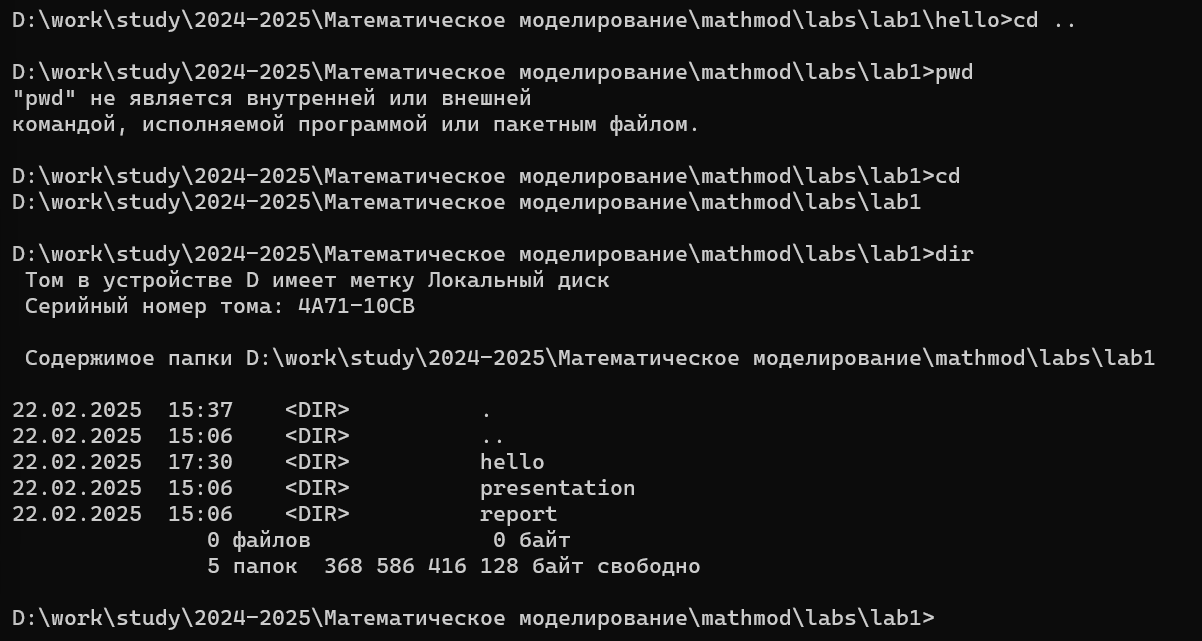


Рис. 87: Клонирование репозиториев

Создадим клон репозитория (рис. 88).

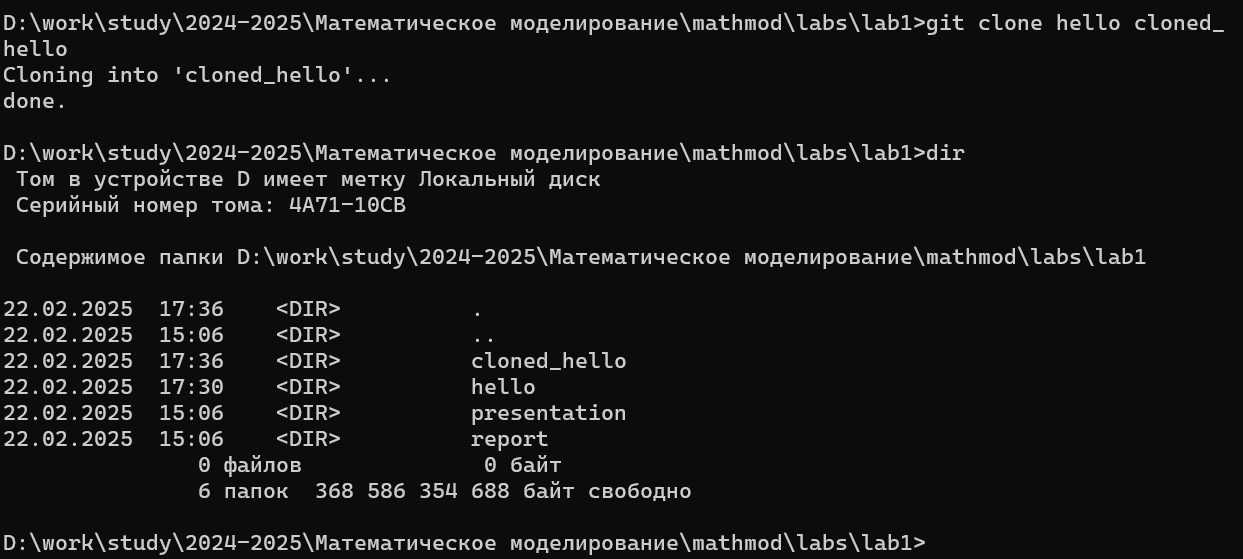


Рис. 88: Создание клона репозитория hello

Давайте взглянем на клонированный репозиторий (рис. 89).

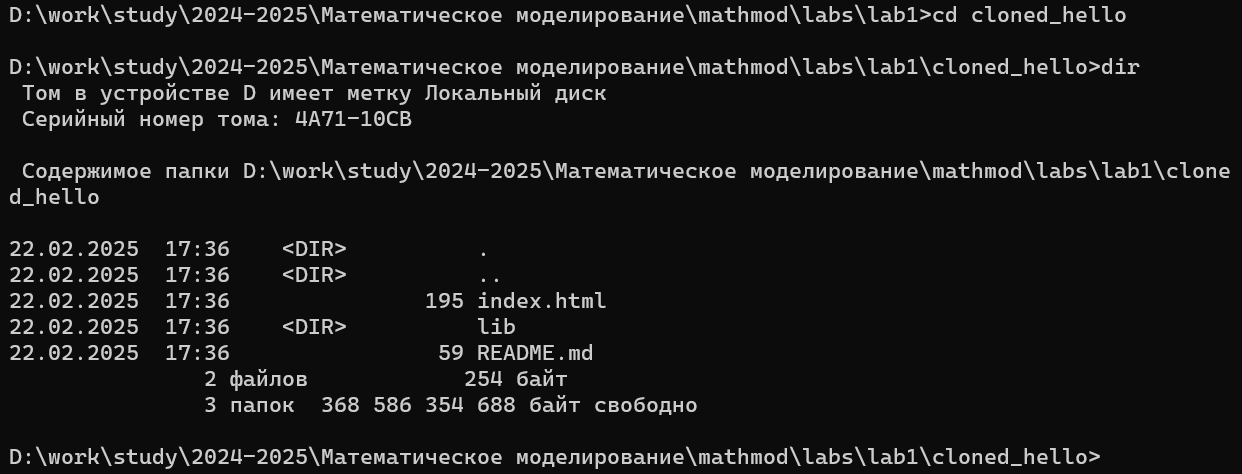


Рис. 89: Просмотр клонированного репозитория

Просмотрим историю репозитория (рис. 90).

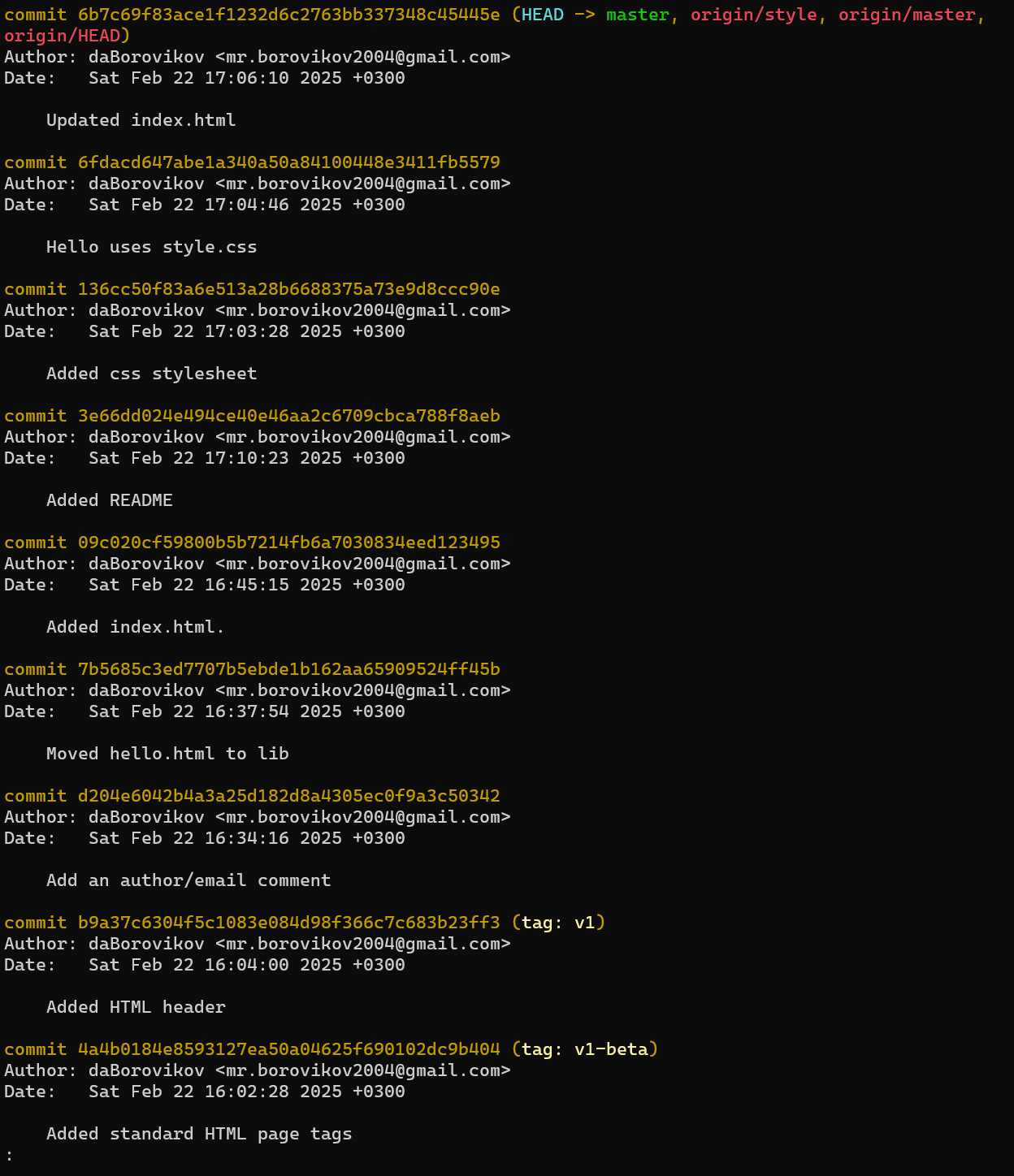


Рис. 90: Просмотр истории репозитория

Мы видим, что клонированный репозиторий знает об имени по умолчанию удаленного репозитория. Давайте посмотрим, можем ли мы получить более подробную информацию об имени по умолчанию (рис. 91).



Рис. 91: Изучение origin

Давайте посмотрим на ветки, доступные в нашем клонированном репозитории (рис. 92).

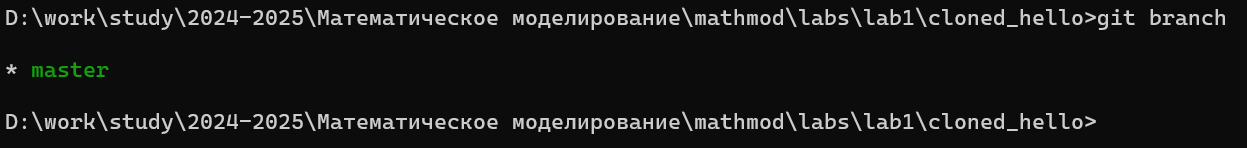


Рис. 92: Просмотр доступных веток в клонированном репозитории

Для того, чтобы увидеть все ветки, попробуем следующую команду (рис. 93).

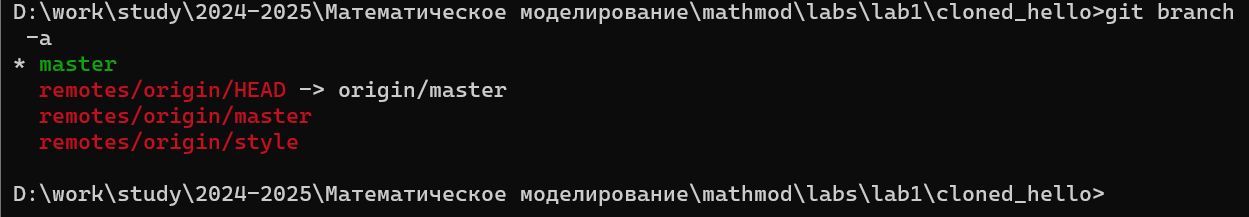


Рис. 93: Просмотр всех веток

Перейдём в репозиторий hello (рис. 94).

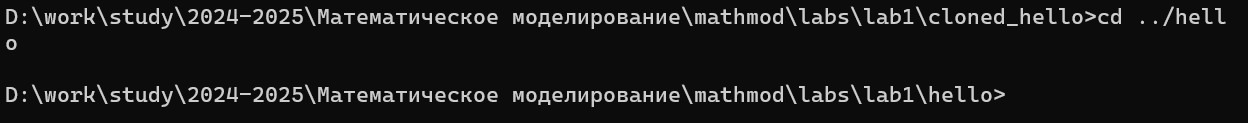


Рис. 94: Переход в репозиторий hello

Внесём изменения в файл README.md (рис. 95).



Рис. 95: Внесение изменения в файл README.md

Добавим это изменение и сделаем коммит (рис. 96).

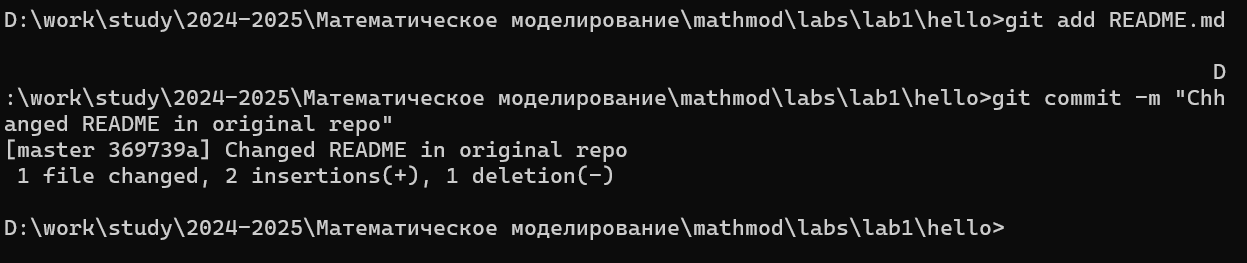


Рис. 96: Добавление изменения и создание коммита

Научимся извлекать изменения из удаленного репозитория (рис. 97).

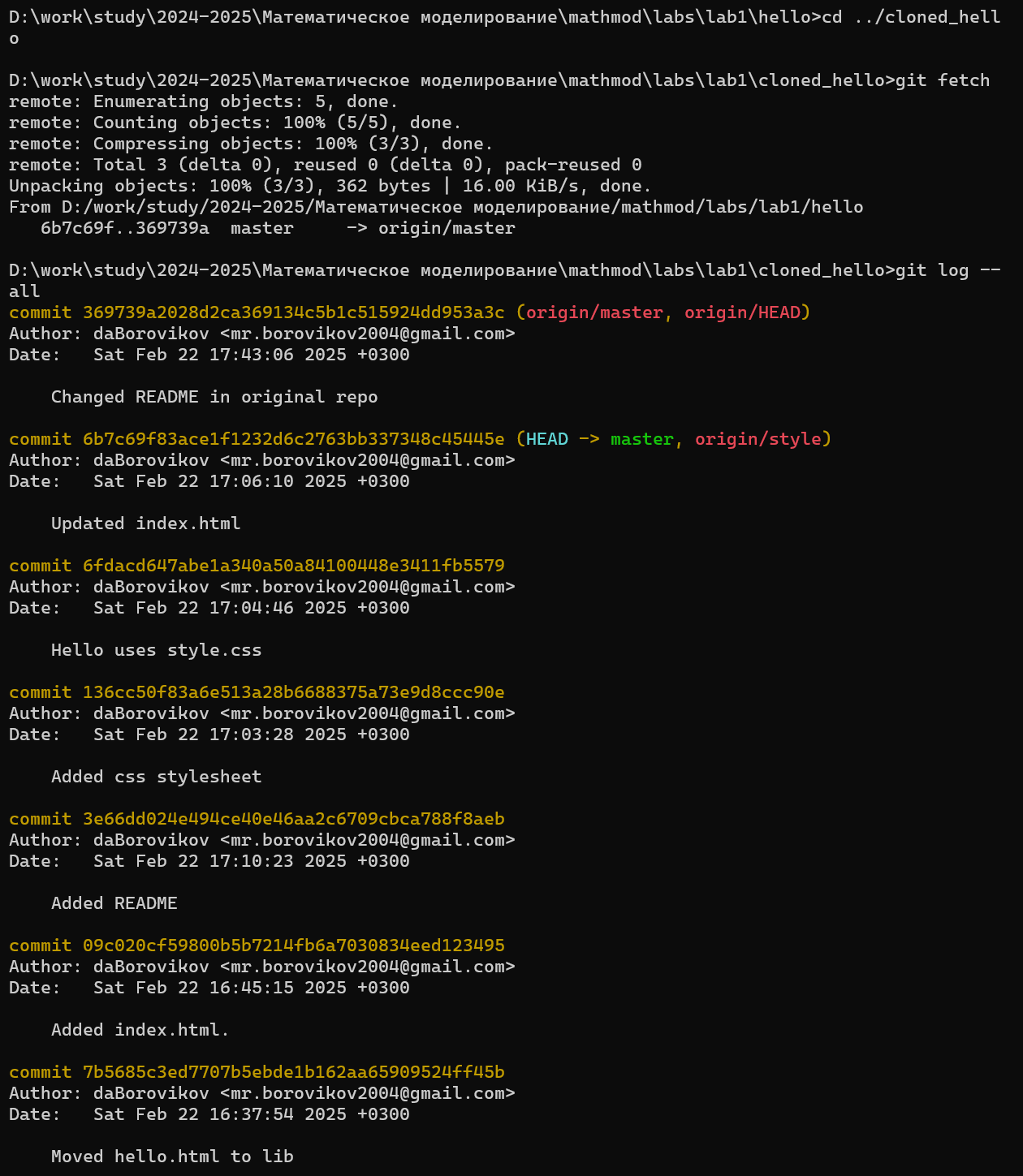


Рис. 97: Извлечение изменений

Мы можем продемонстрировать, что клонированный файл README.md не изменился (рис. 98).

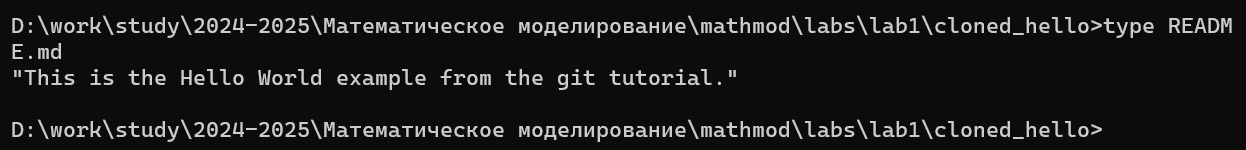


Рис. 98: Проверка README.md

Сольём извлеченные изменения в локальную ветку master (рис. 99).

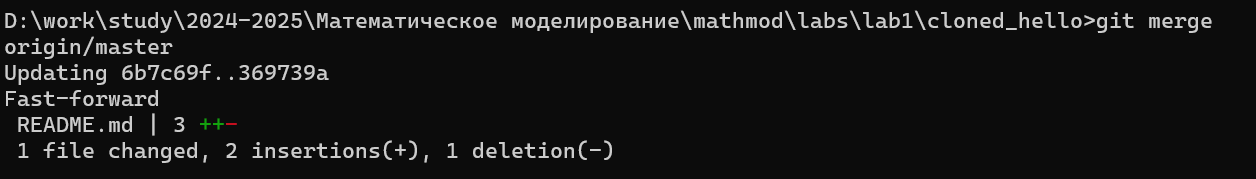


Рис. 99: Слив извлеченных изменений в локальную ветку master

Теперь давайте рассмотрим объединение fetch и merge в одну команду (рис. 100).

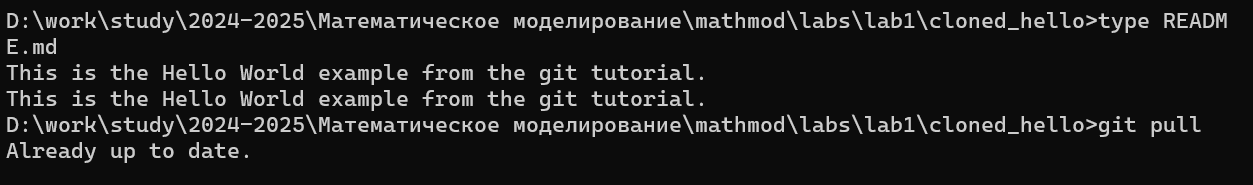


Рис. 100: Объединение fetch и merge в одну команду

Добавим локальную ветку, которая отслеживает удаленную ветку (рис. 101).

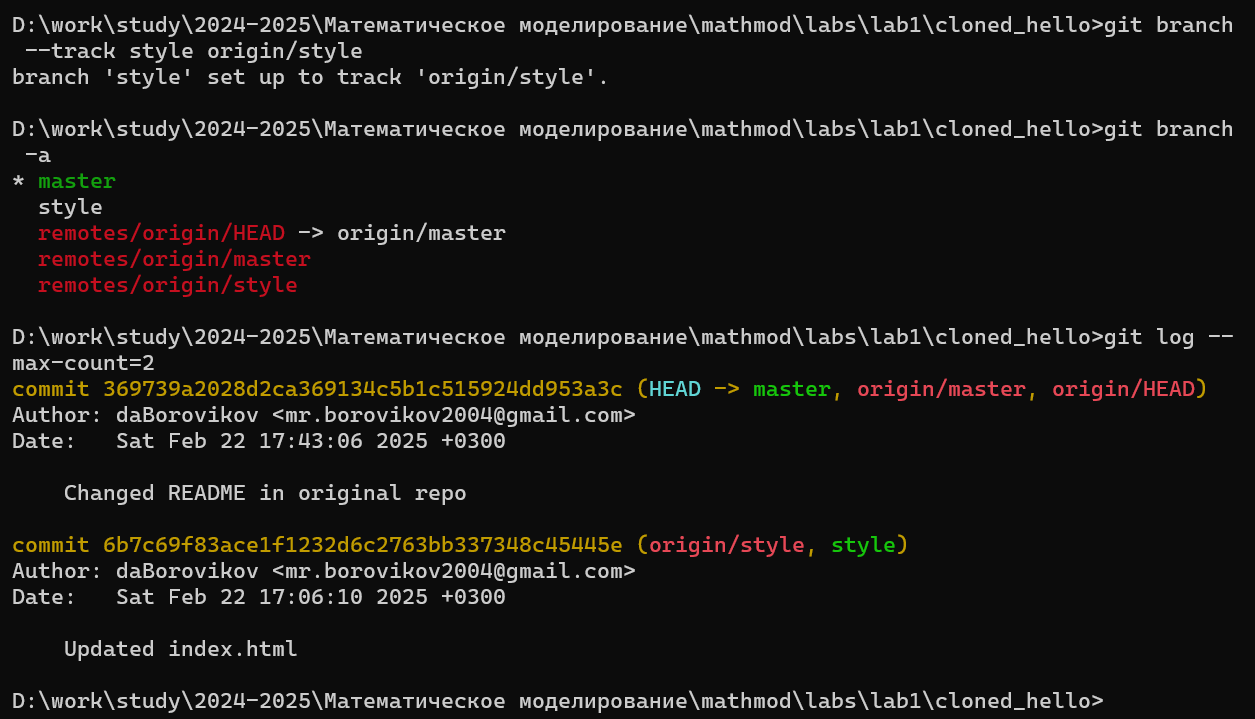


Рис. 101: Добавление локальной ветки, которая отслеживает удаленную ветку

Создадим чистый репозиторий (рис. 102).

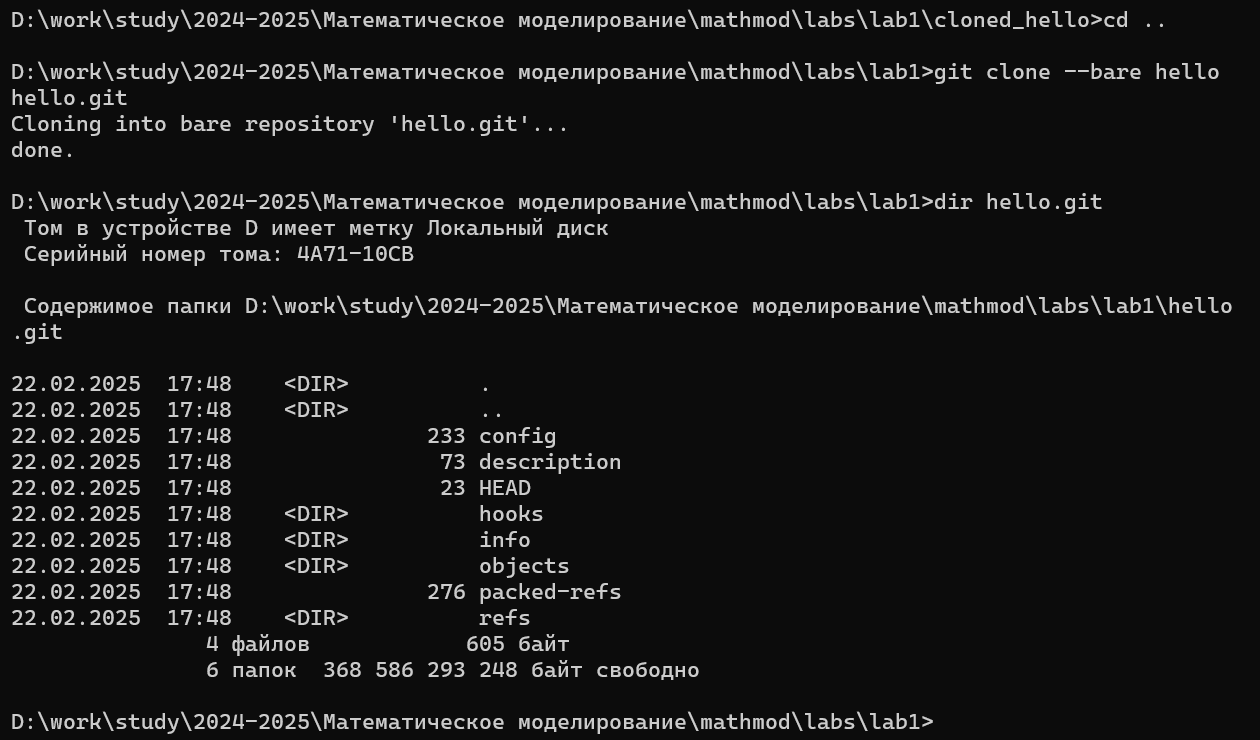


Рис. 102: Создание чистого репозитория

Добавим удаленный репозиторий (рис. 103).

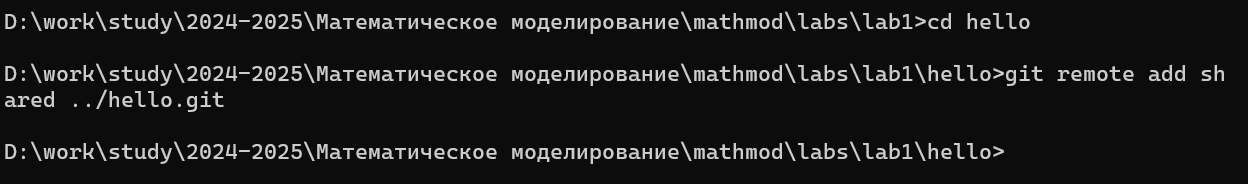


Рис. 103: Добавление удаленного репозитория

Отредактируем файл README.md (рис. 104).

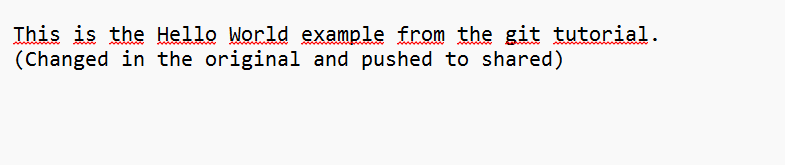


Рис. 104: Редактирование файла README.md

Сделаем коммит и отправим изменения в общий репозиторий (рис. 105).



Рис. 105: Коммит и отправка изменений в общий репозиторий

Научимся извлекать изменения из общего репозитория. Быстро переключимся в клонированный репозиторий и извлечём изменения, только что отправленные в общий репозиторий (рис. 106).

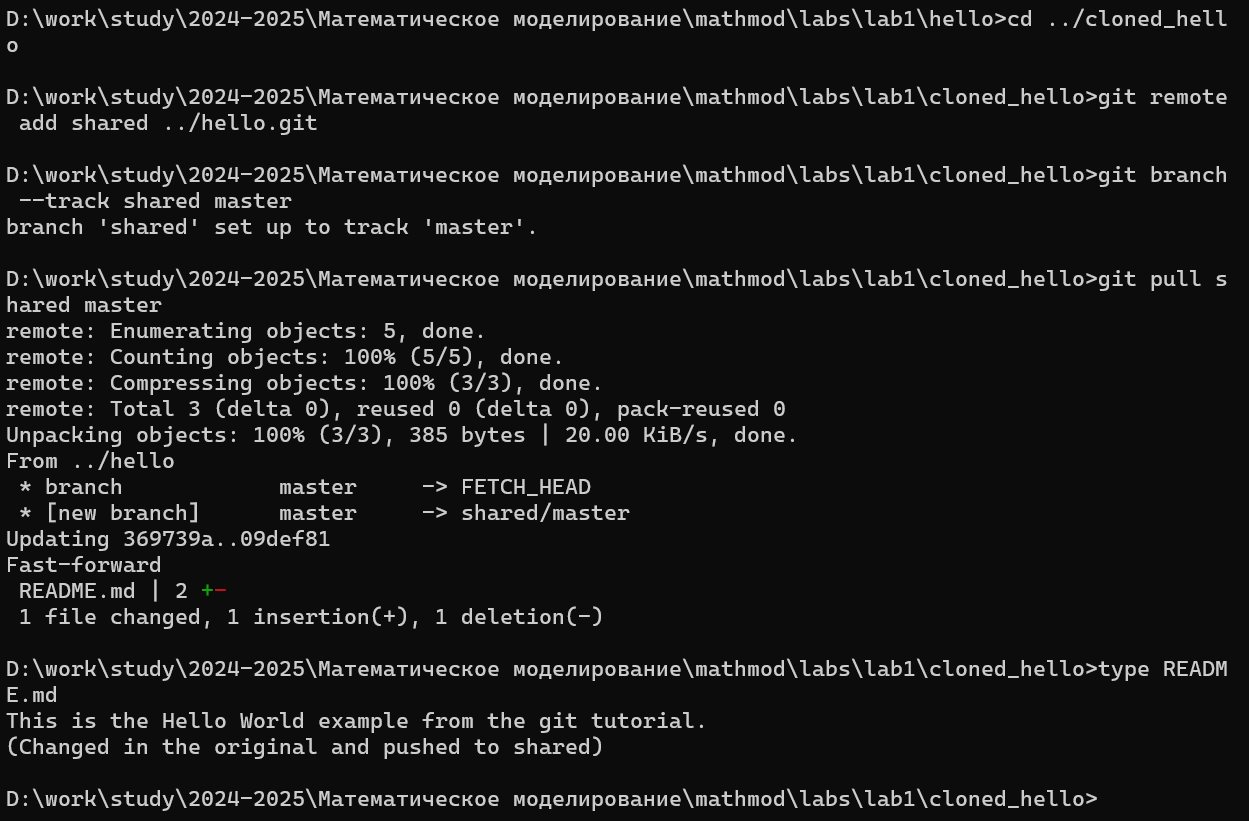


Рис. 106: Извлечение общих изменений

# 3 Выводы

Я приобрел навыки работы с git и разобрал основные его особенности.