Отчет по лабораторной работе №1

Использование git

Бармина Ольга Константиновна

2022 Feb 10th

Содержание

[Цель работы 1](#_Toc95415438)

[Задание 1](#_Toc95415439)

[Теоретическое введение 1](#_Toc95415440)

[Выполнение лабораторной работы 2](#_Toc95415441)

[Выводы 22](#_Toc95415442)

[Список литературы 22](#_Toc95415443)

# Цель работы

Целью данной работы является изучение основ работы с git репозиториями.

# Задание

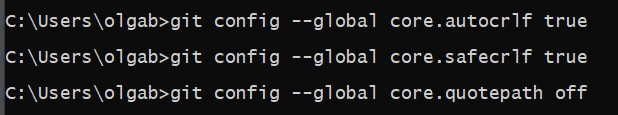
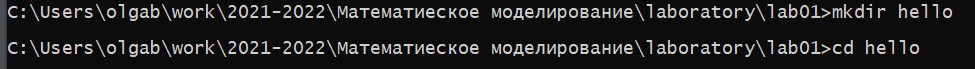
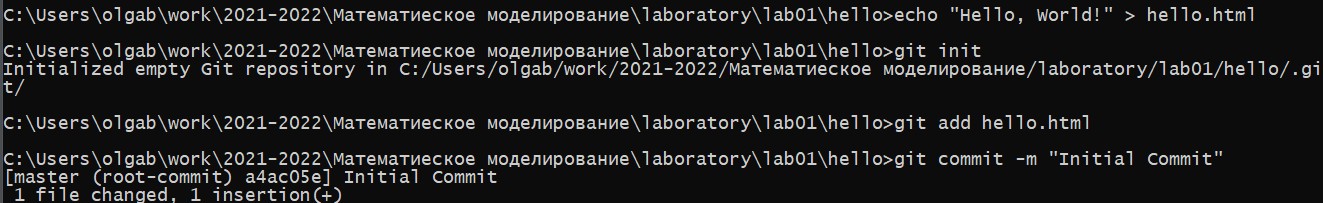
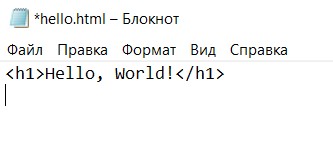
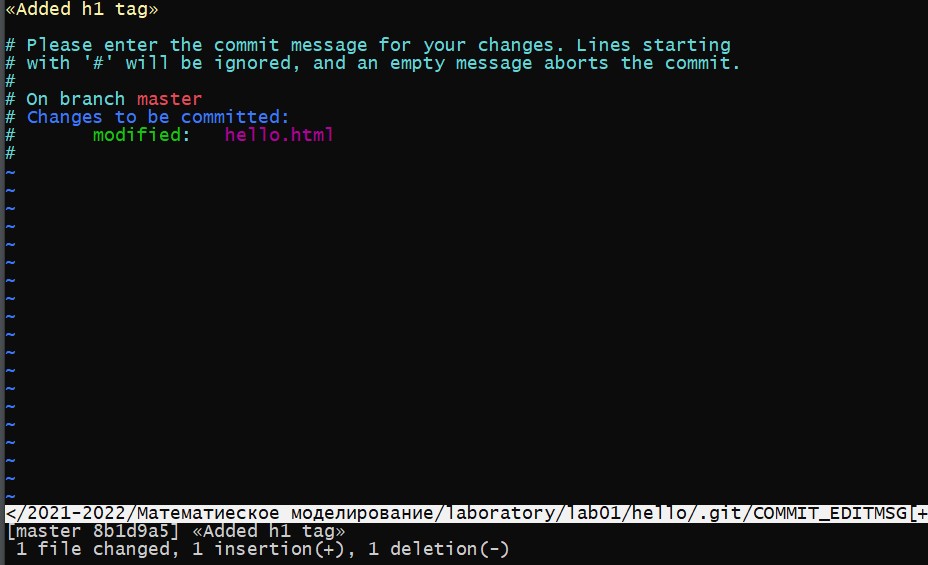
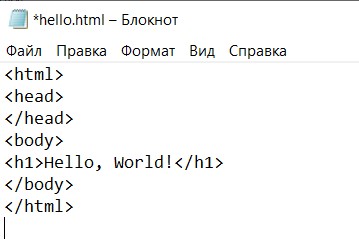
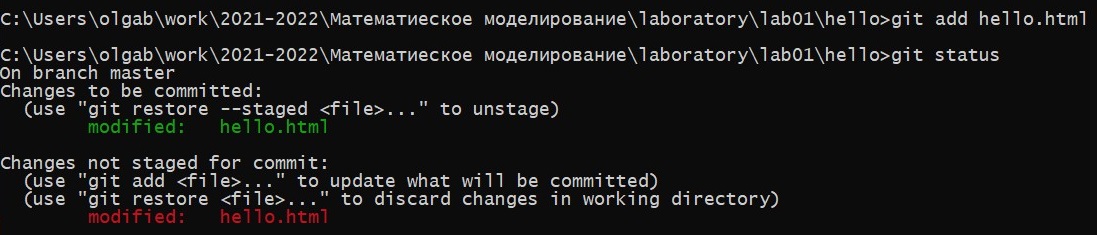
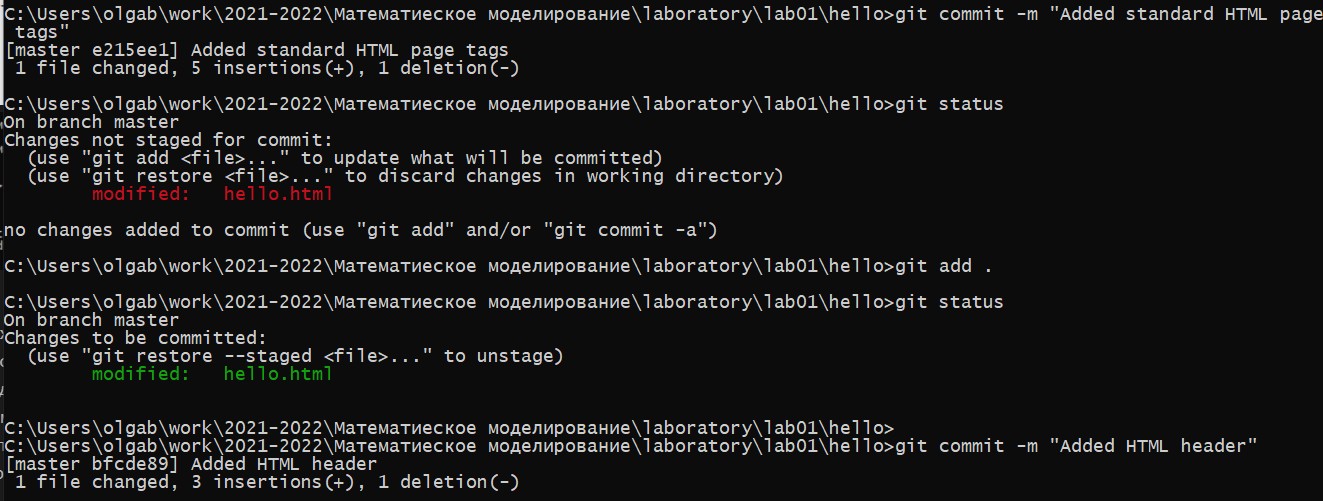
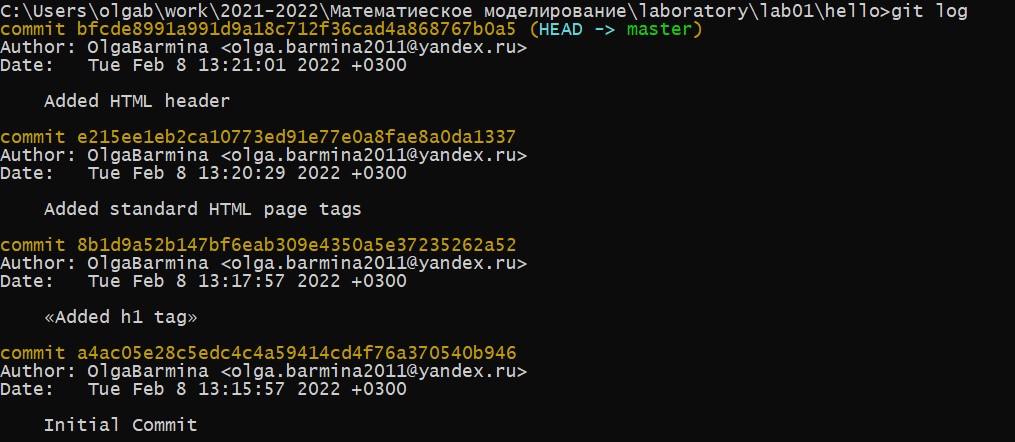
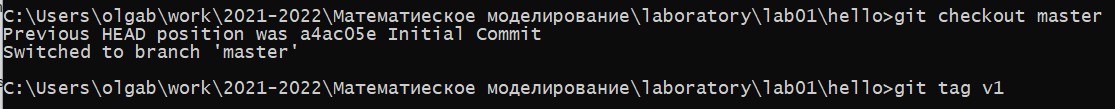
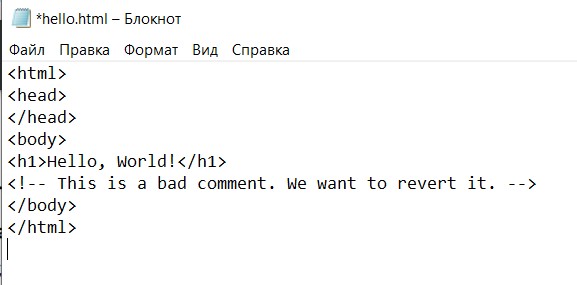
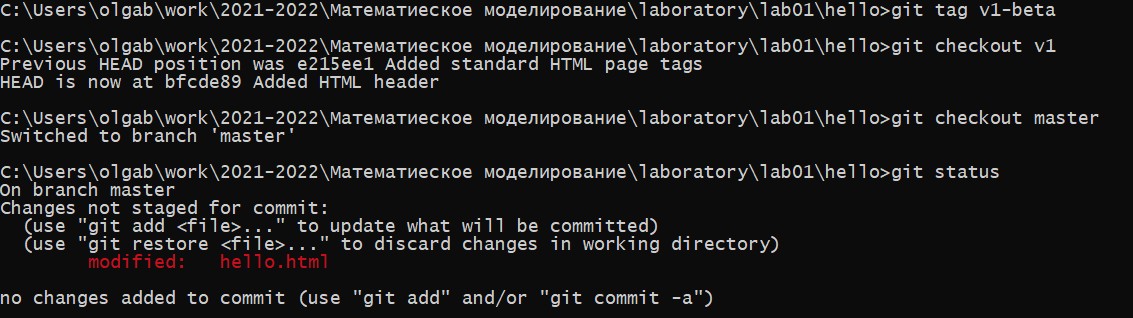
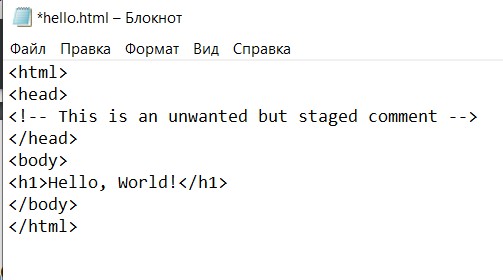
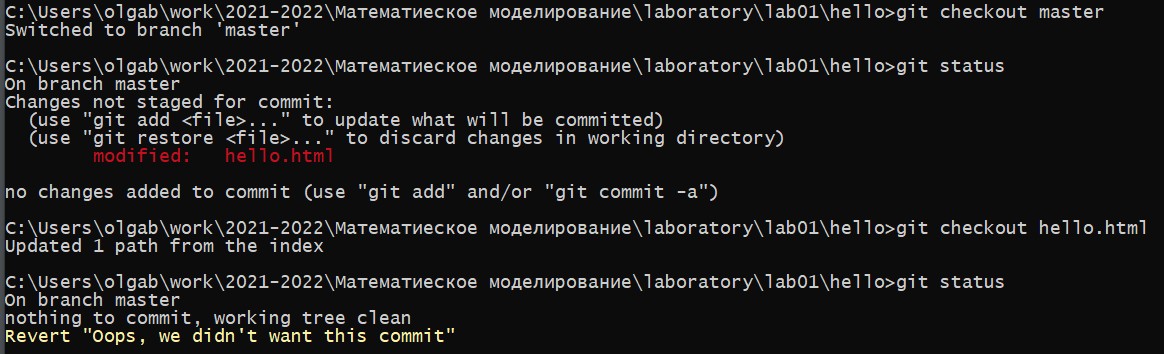
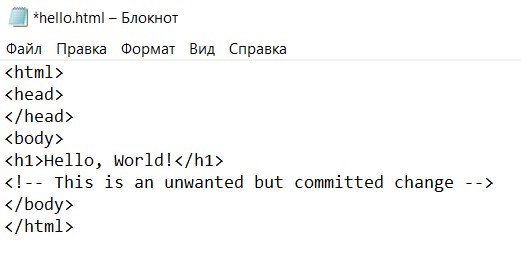
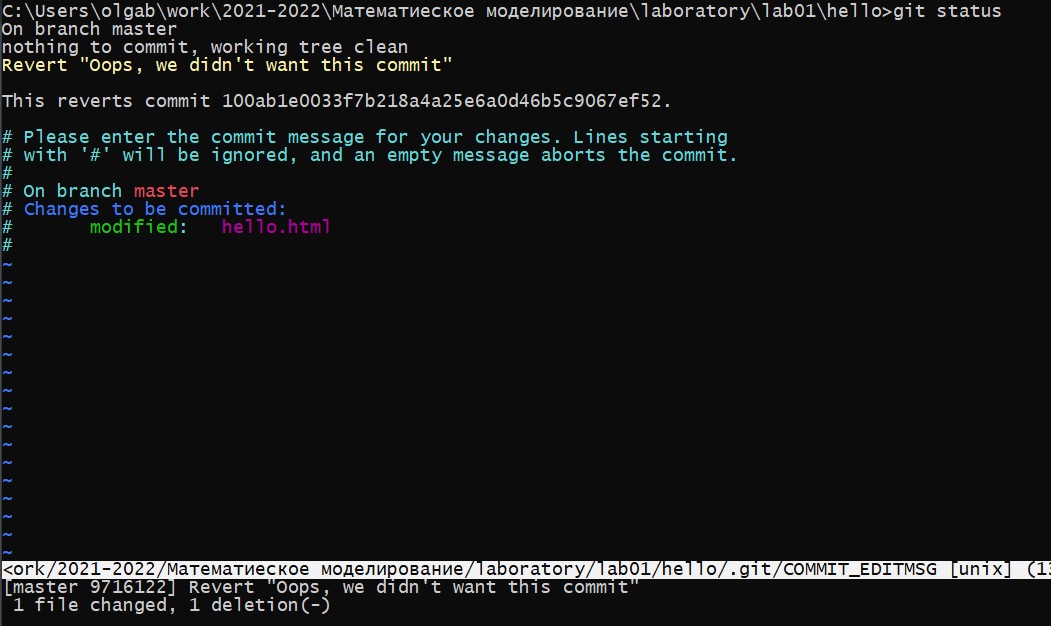
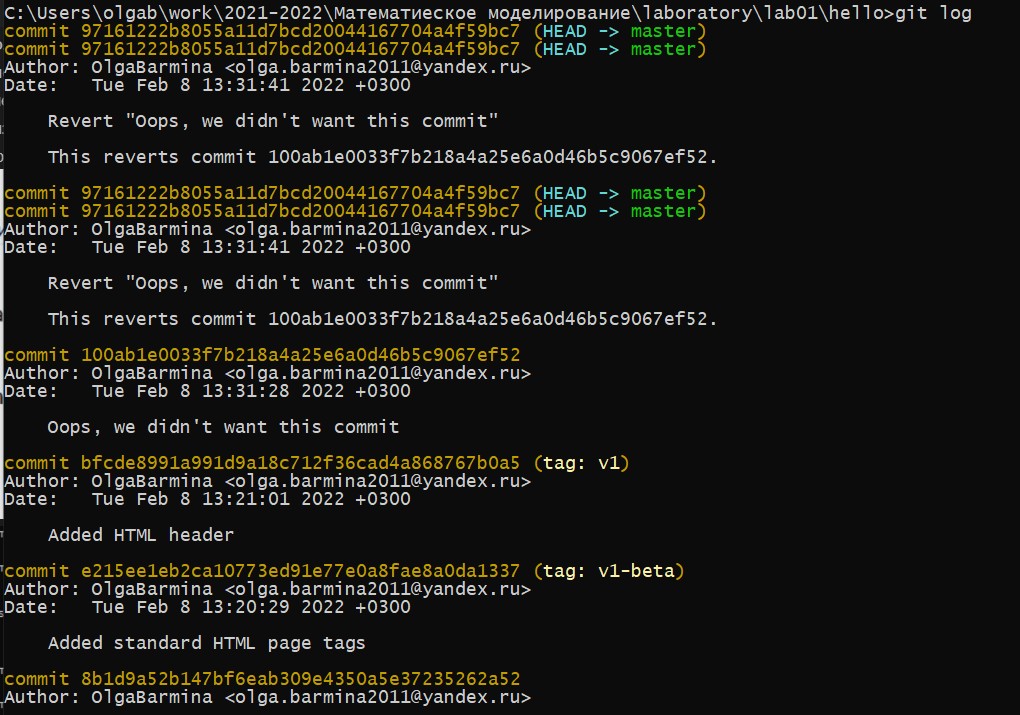
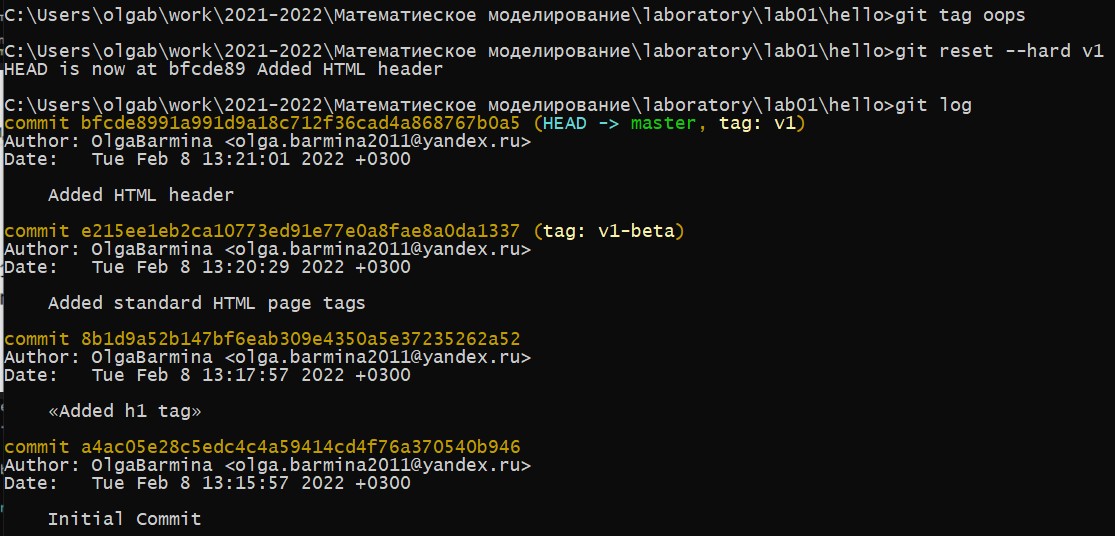
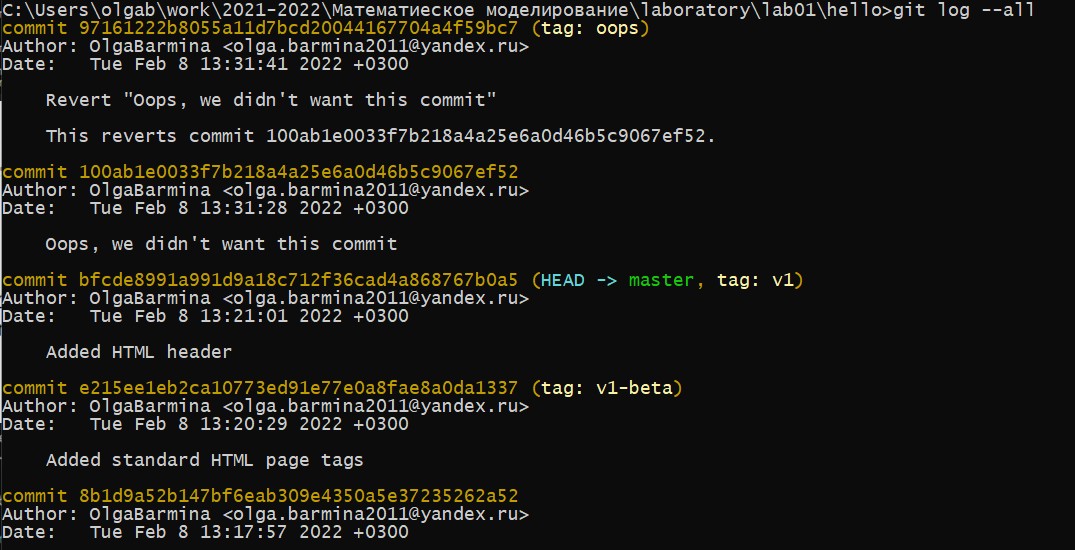
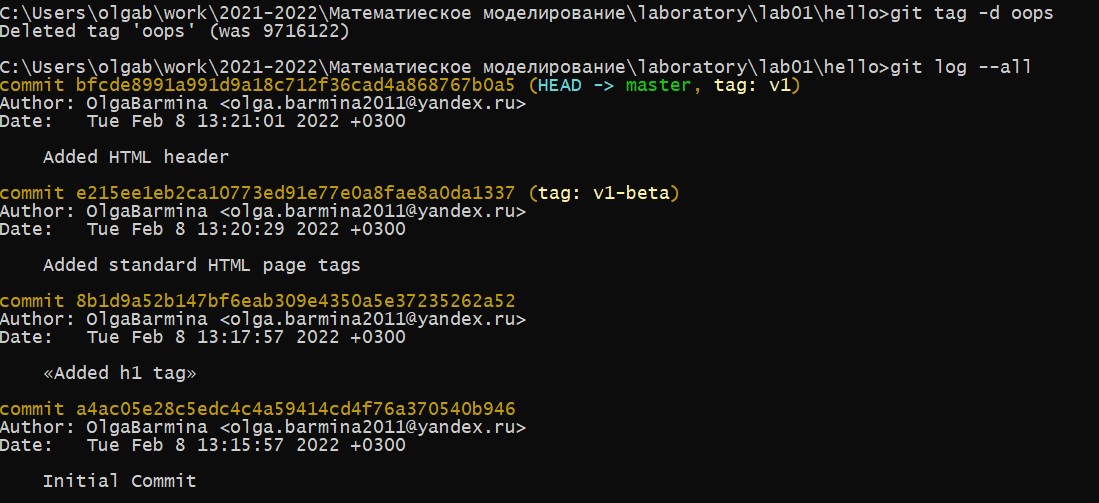
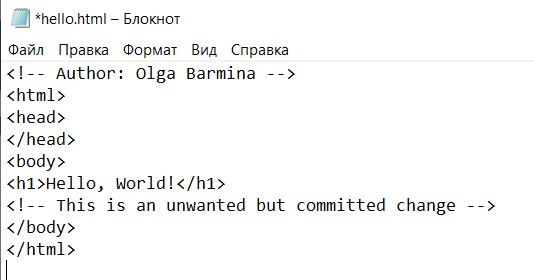
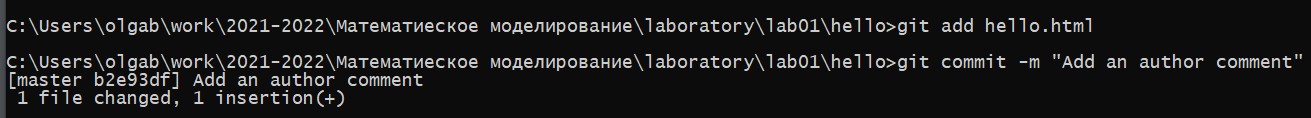
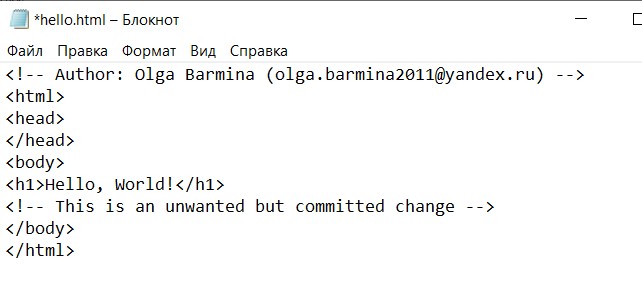
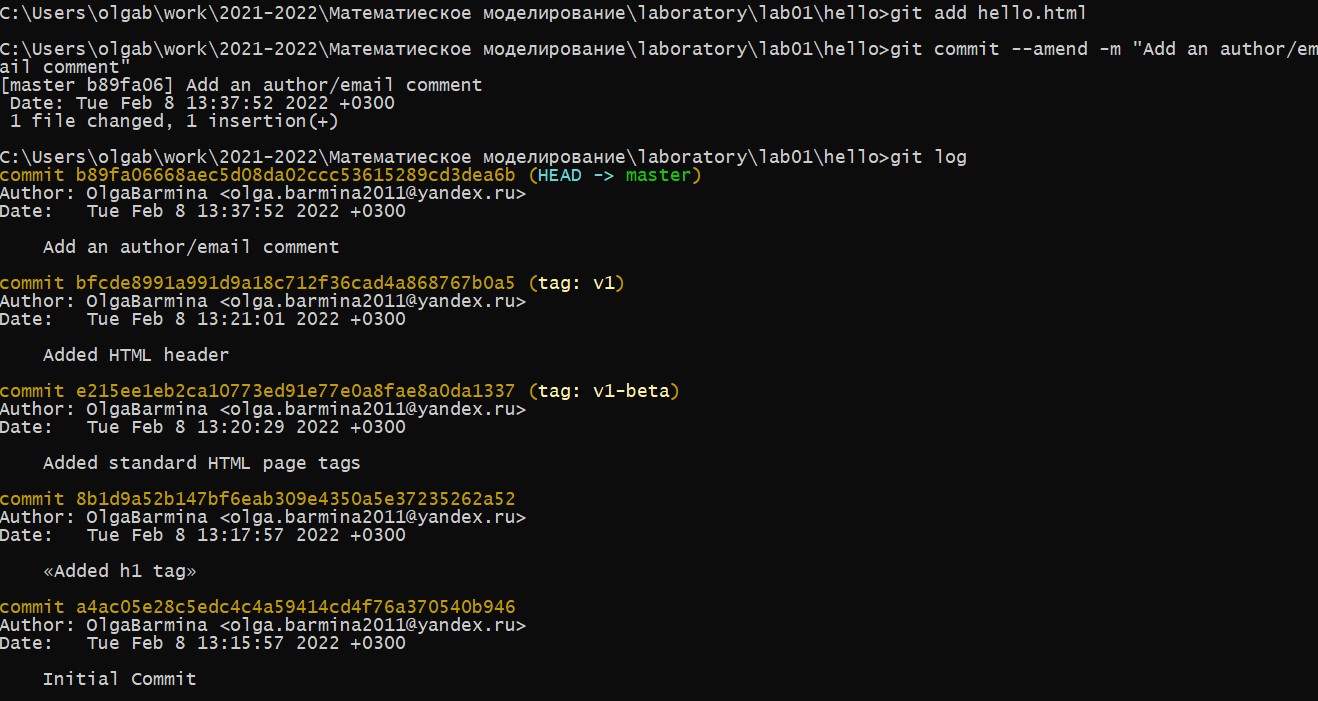
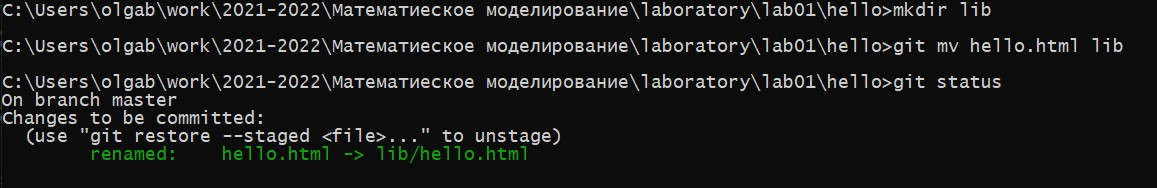
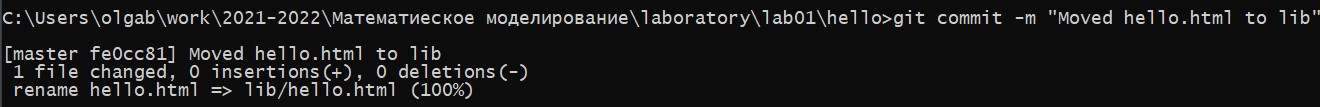
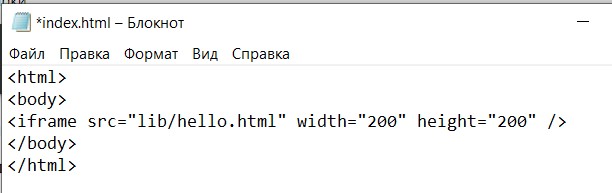
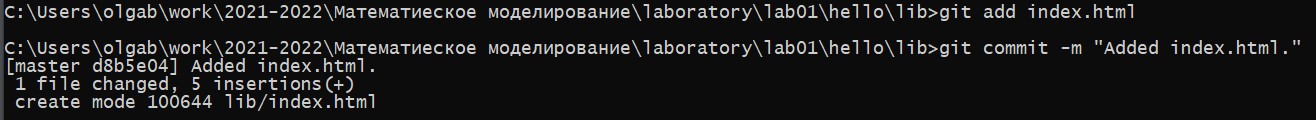
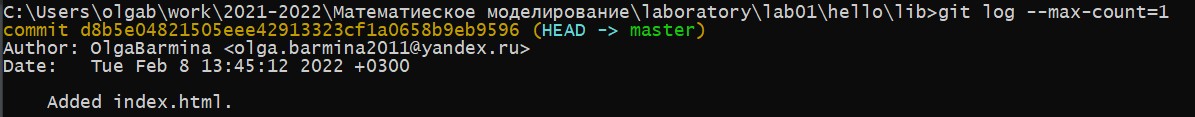
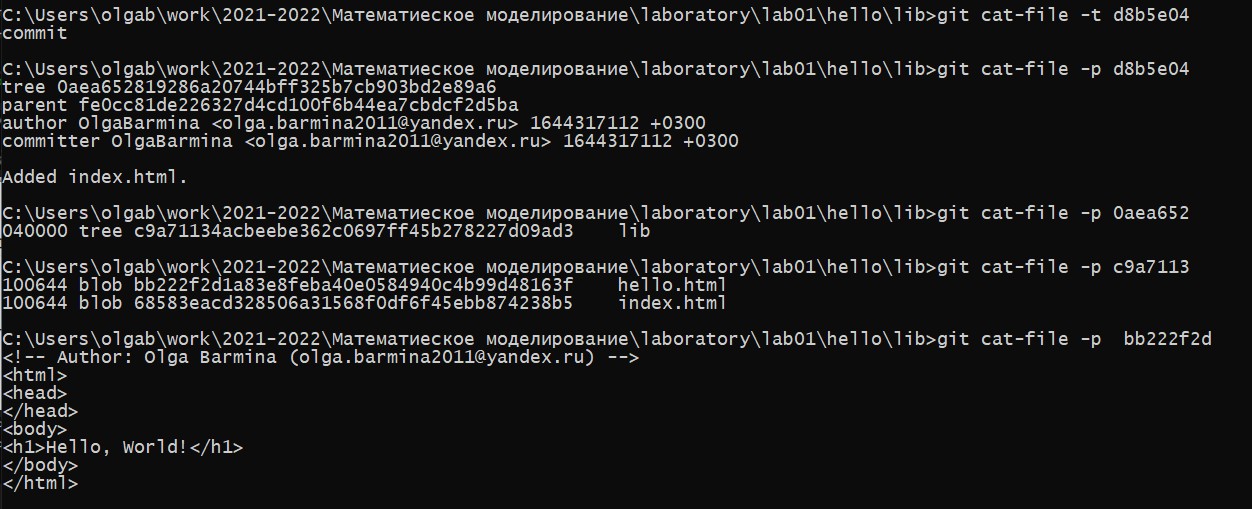
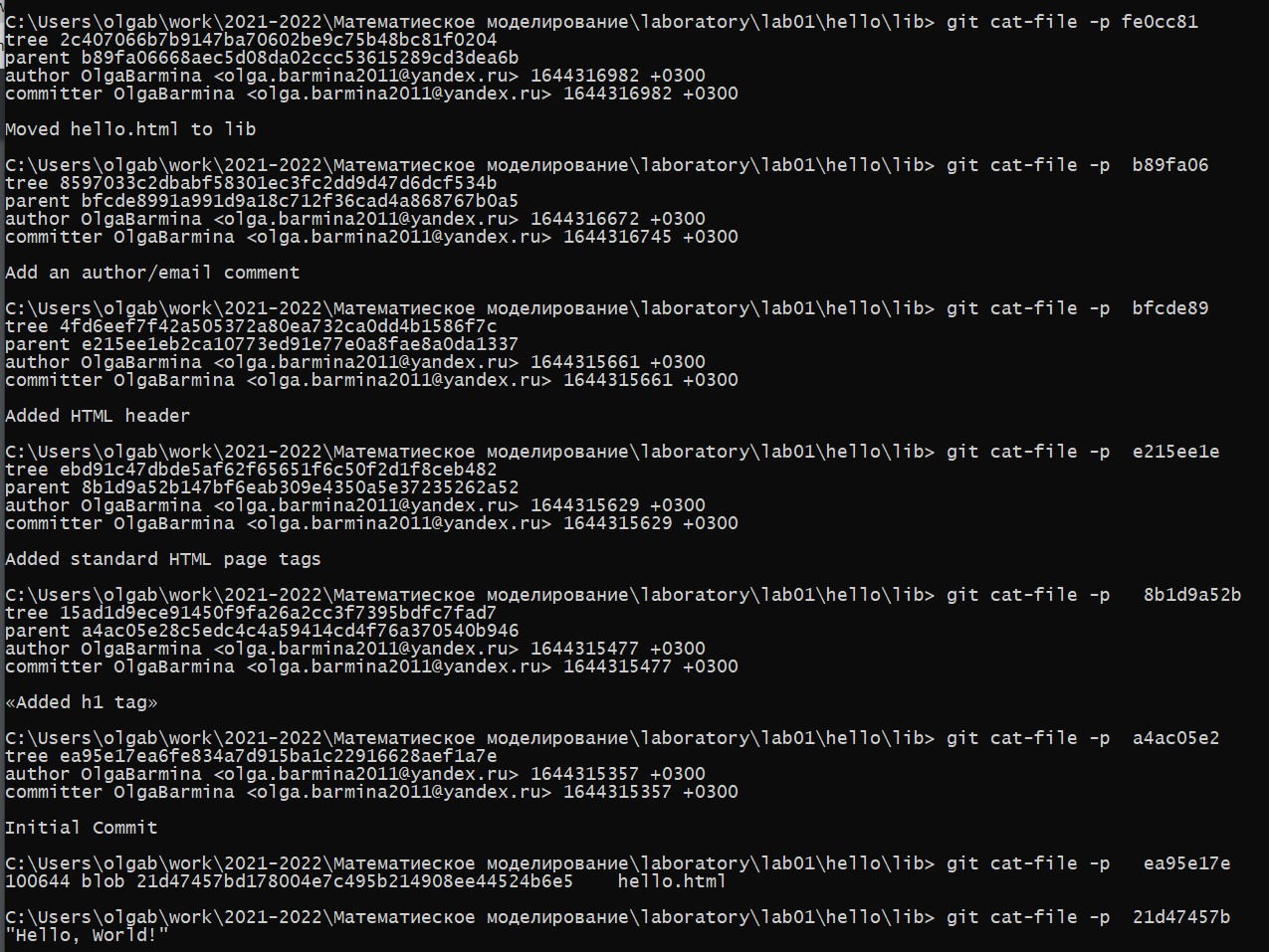
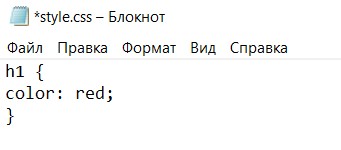
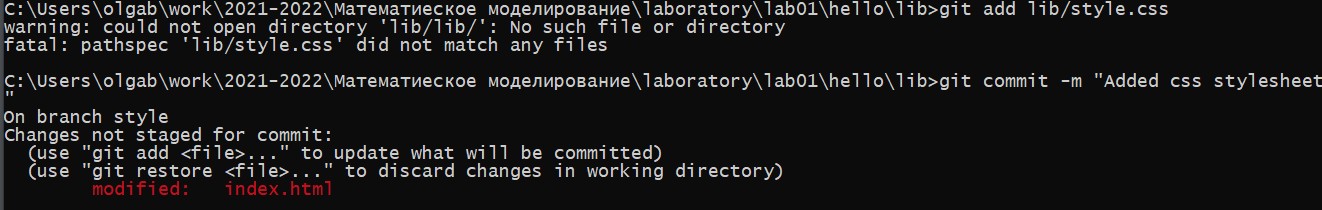
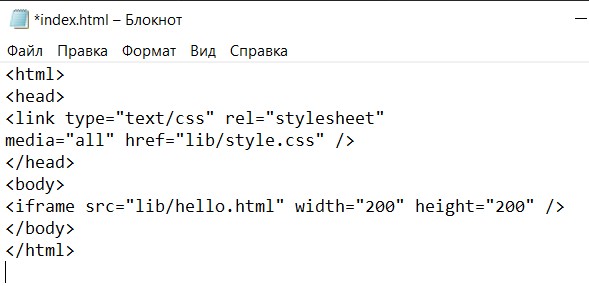
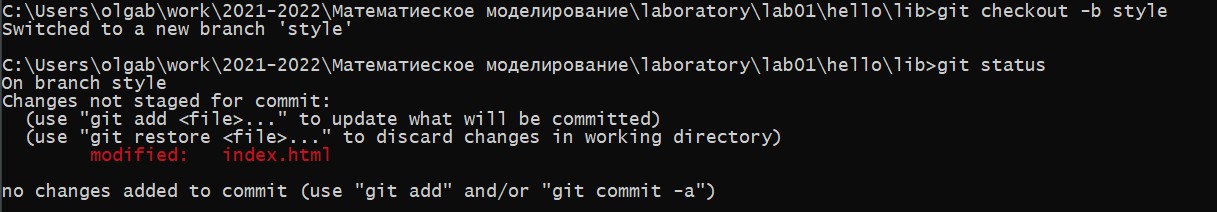
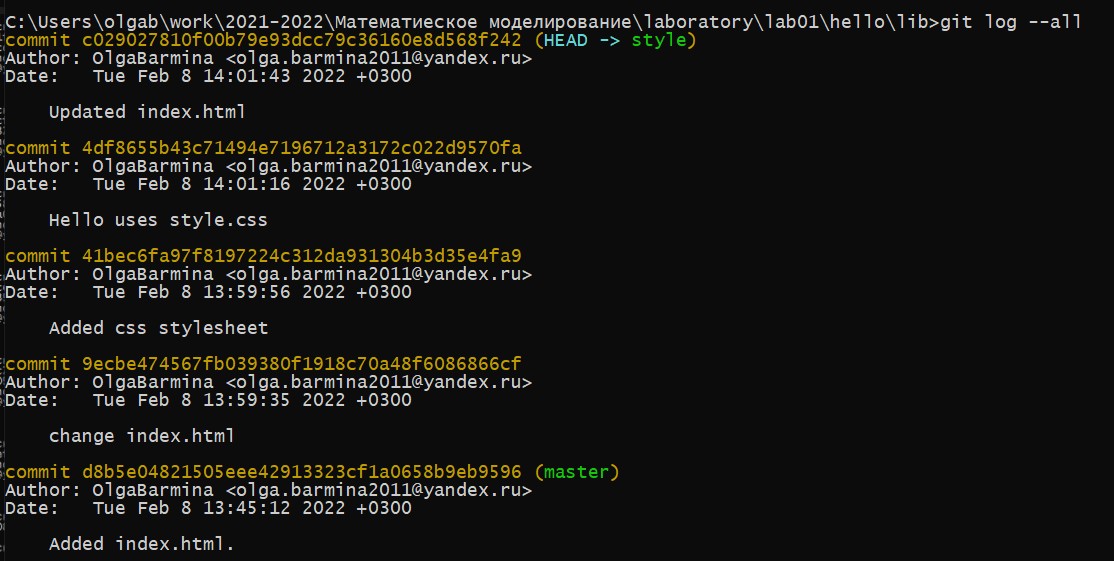
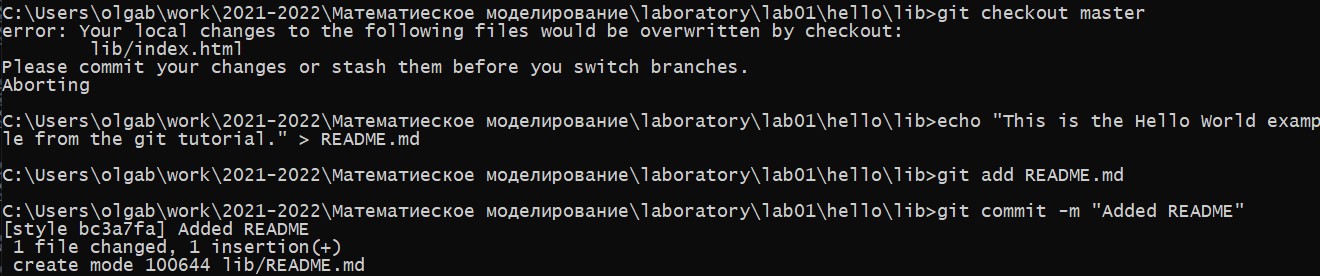
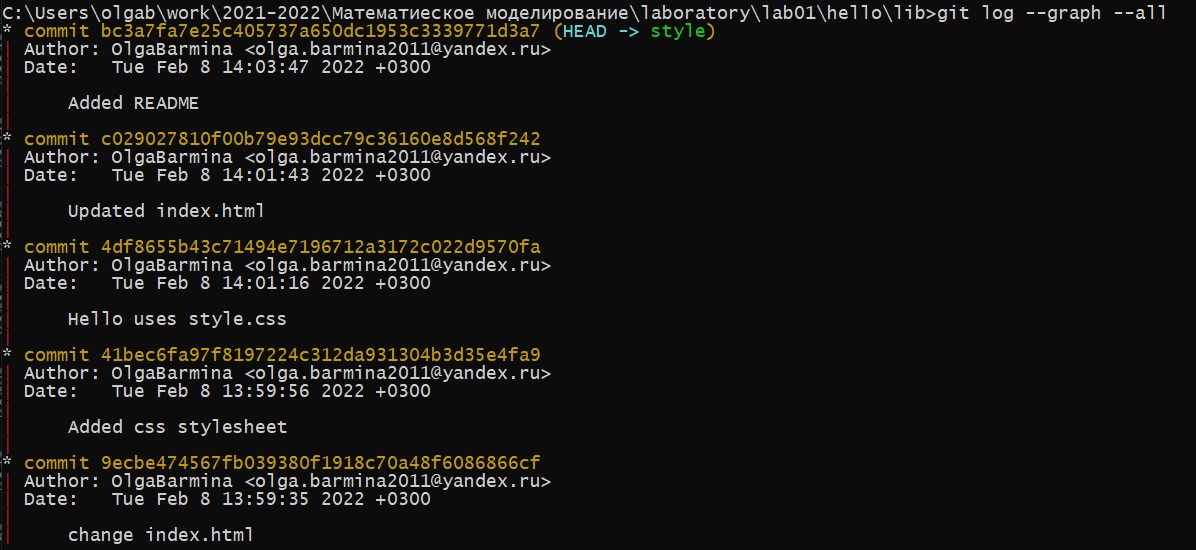
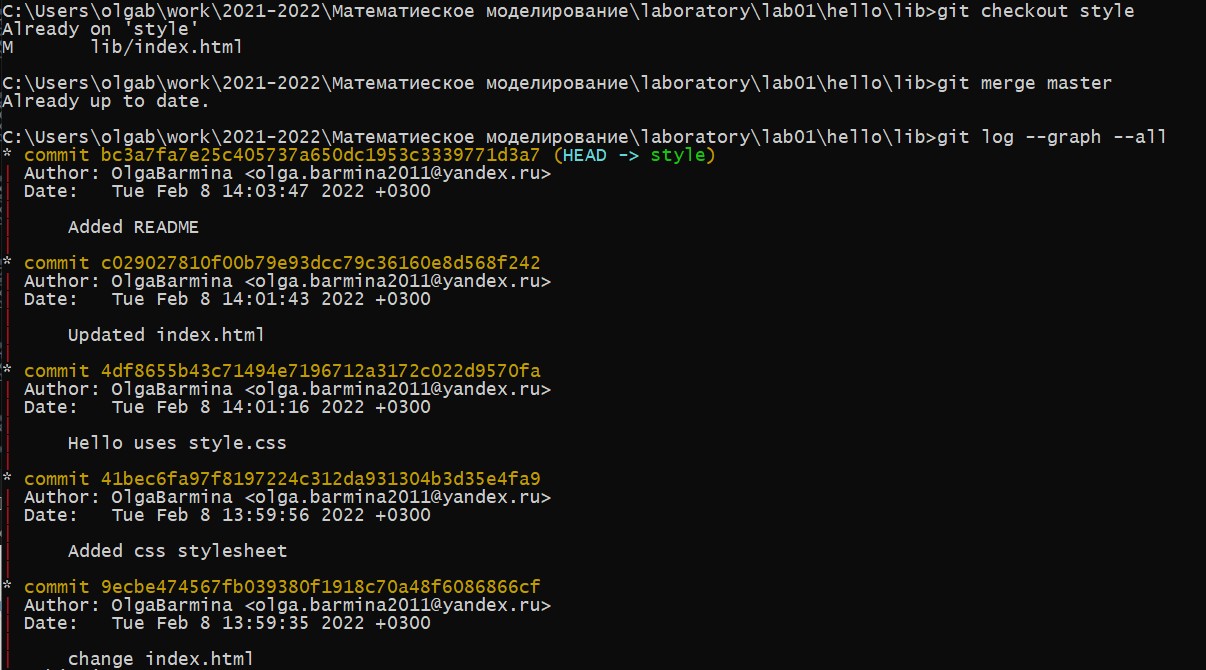
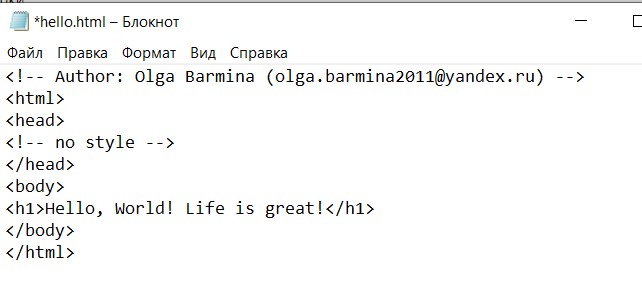
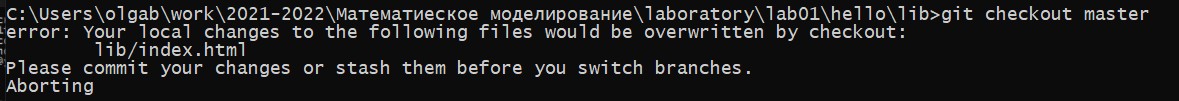
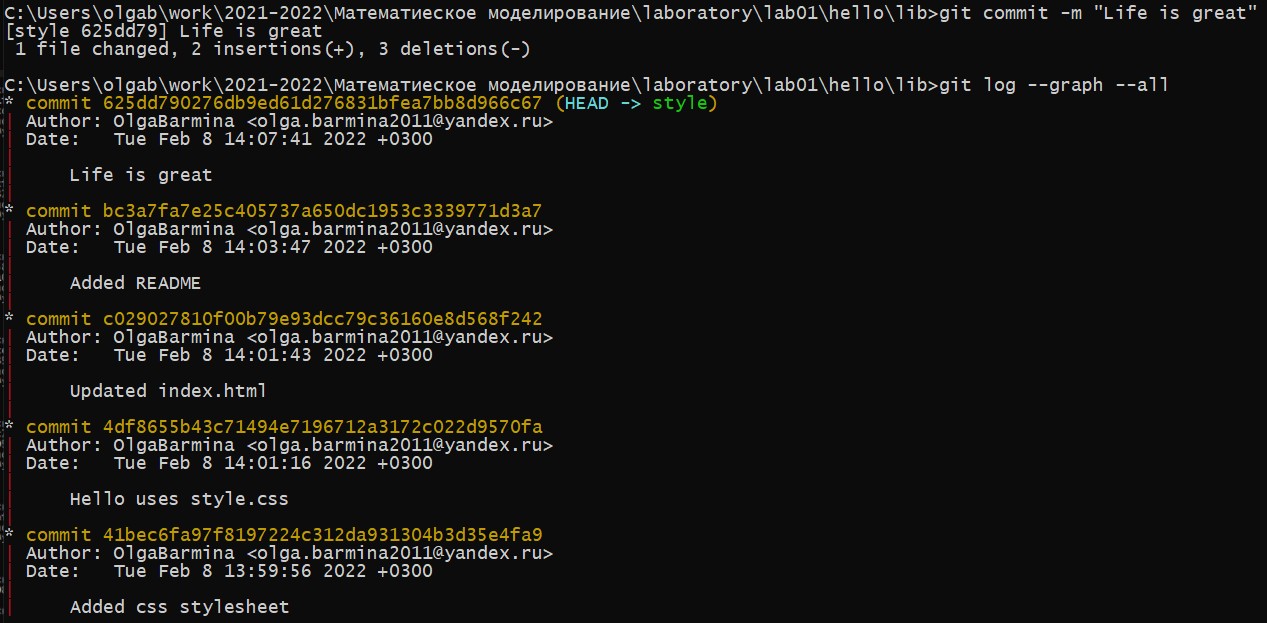
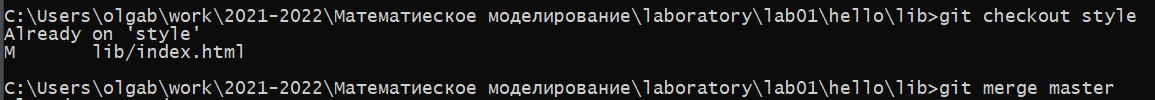
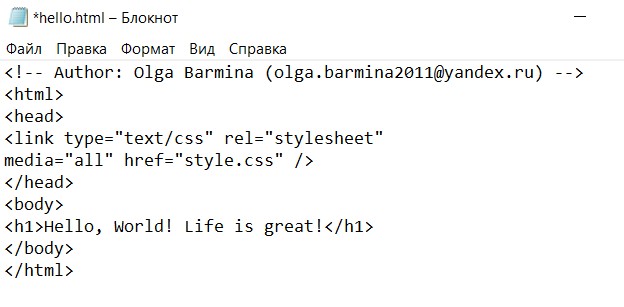
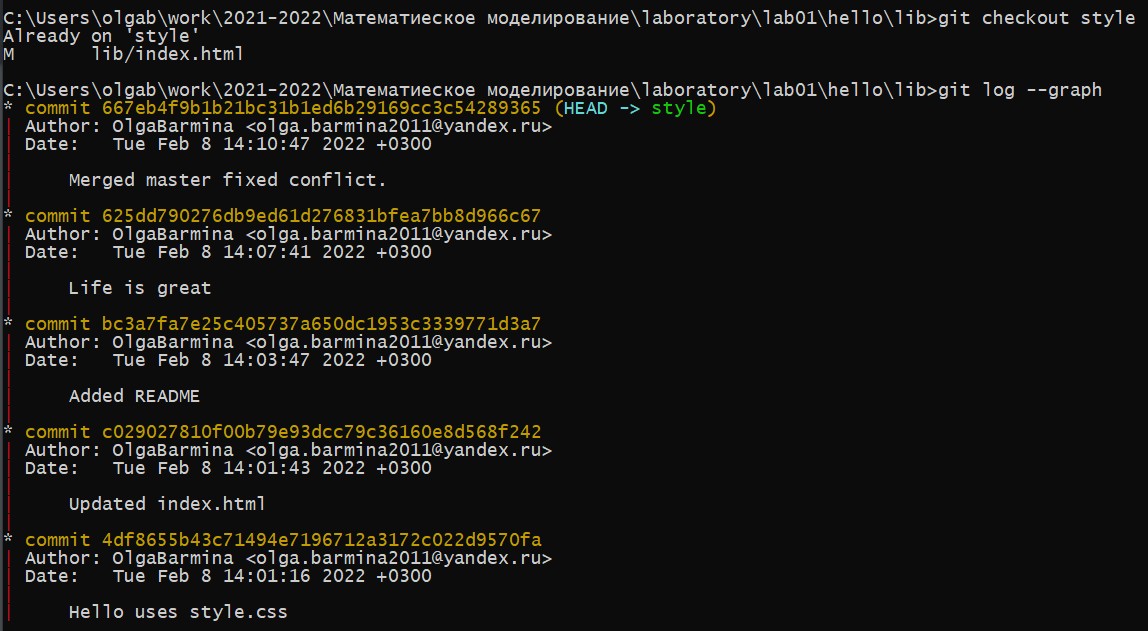
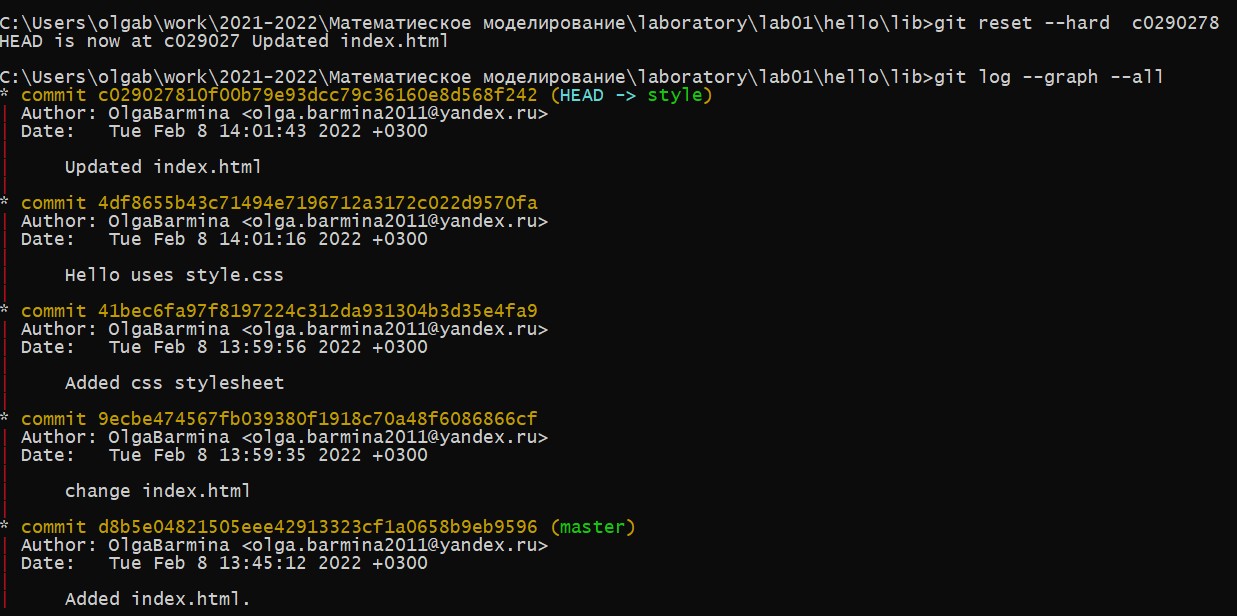
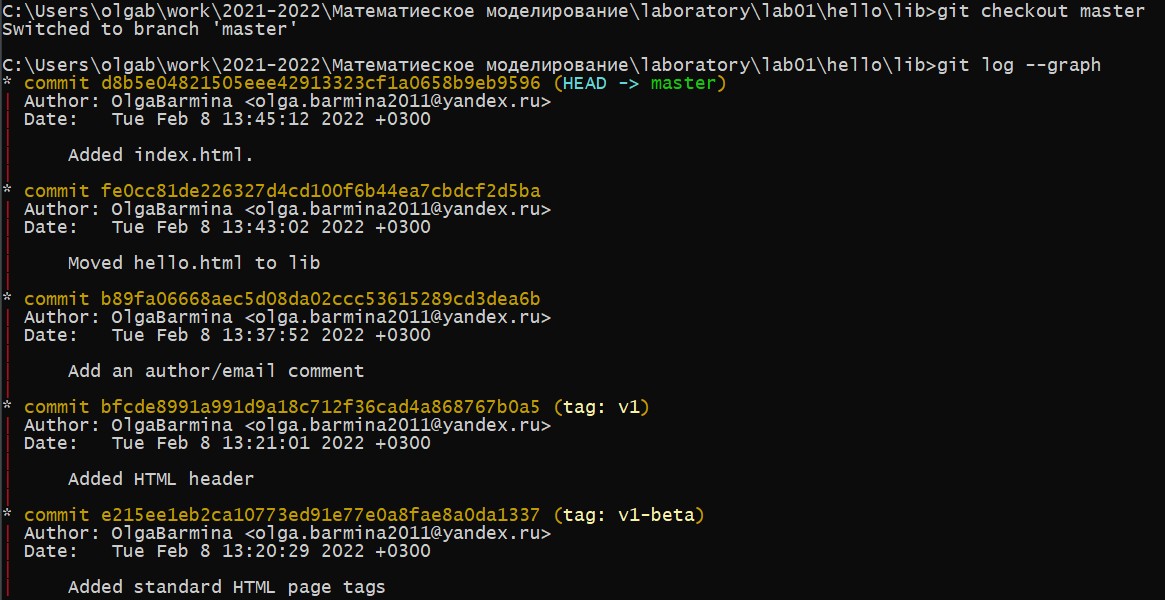
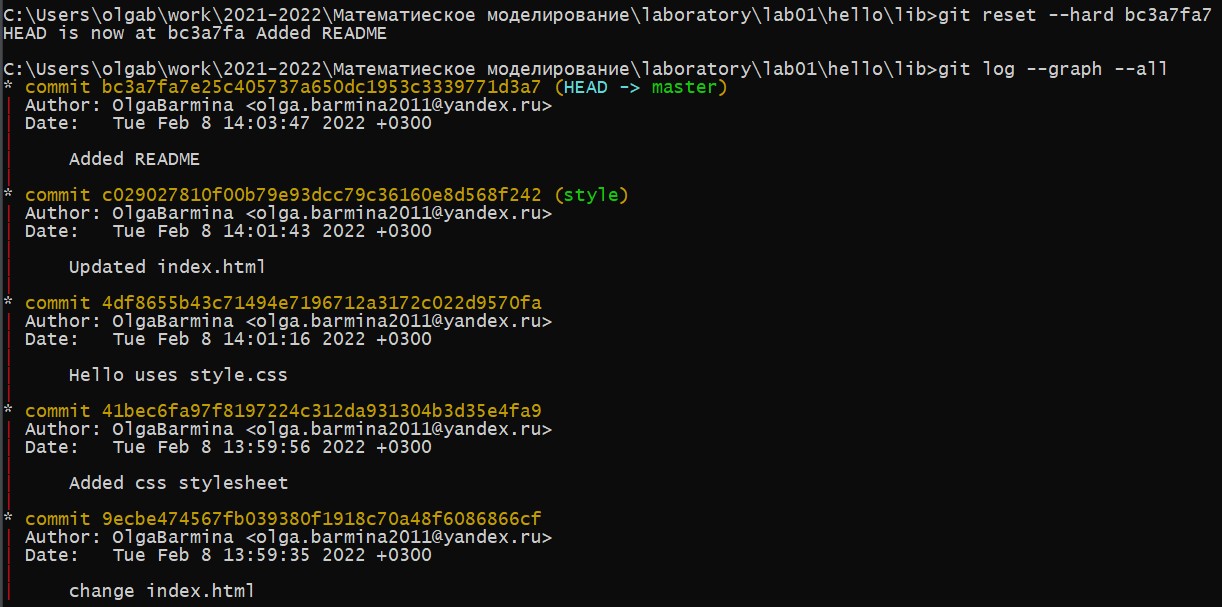
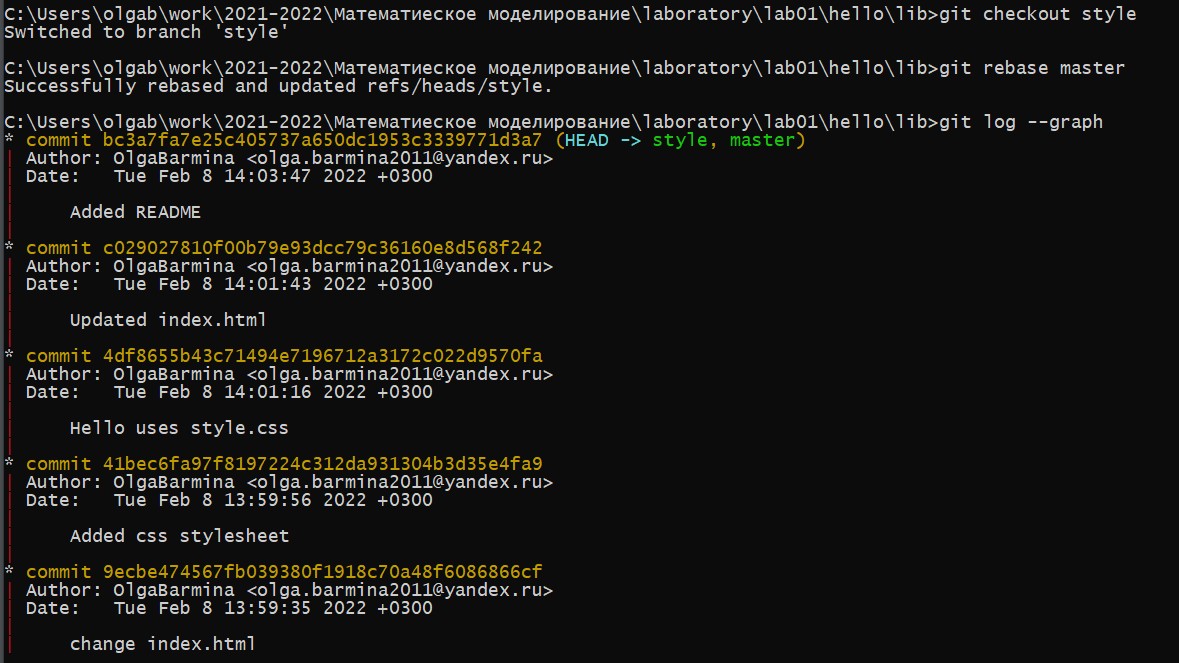
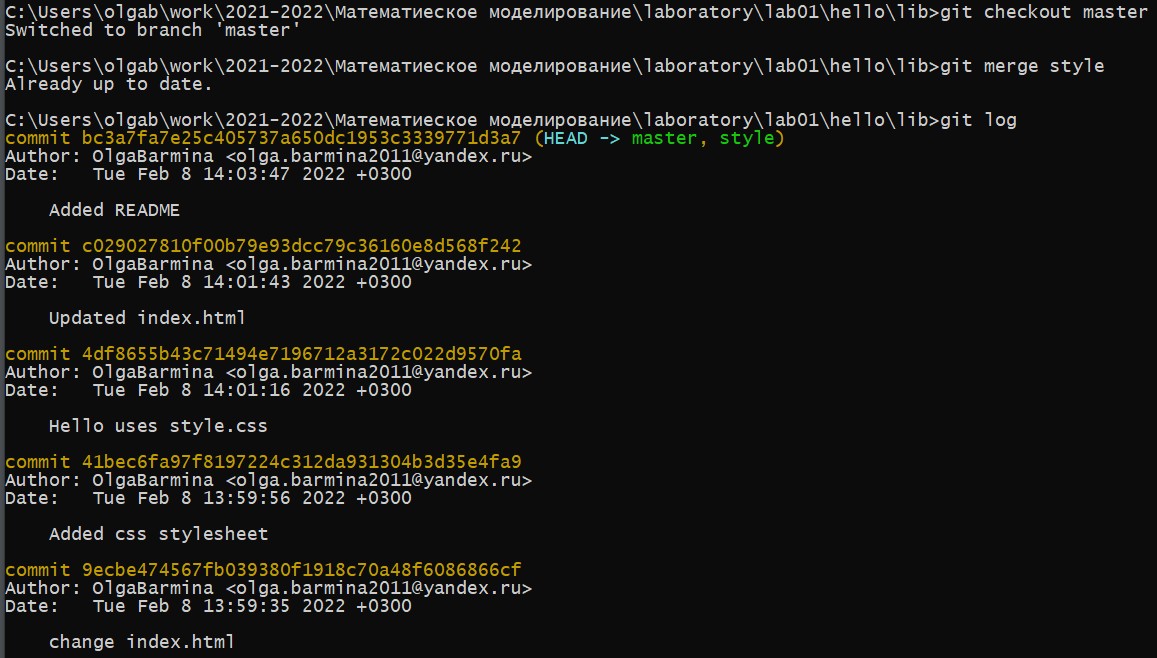
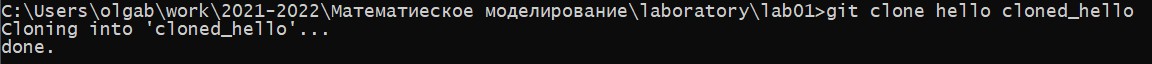
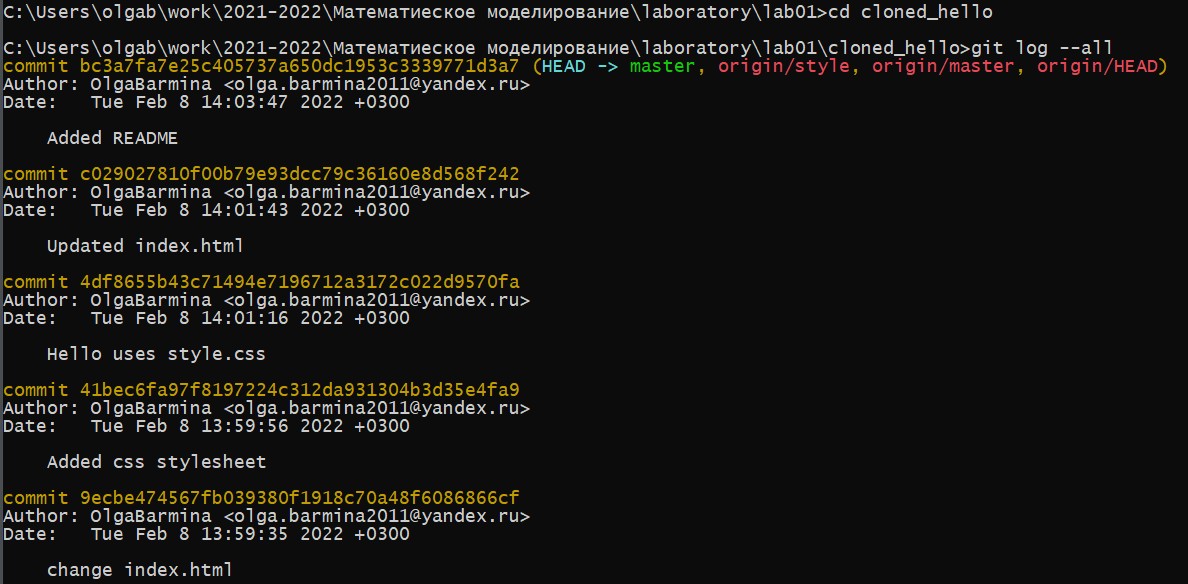
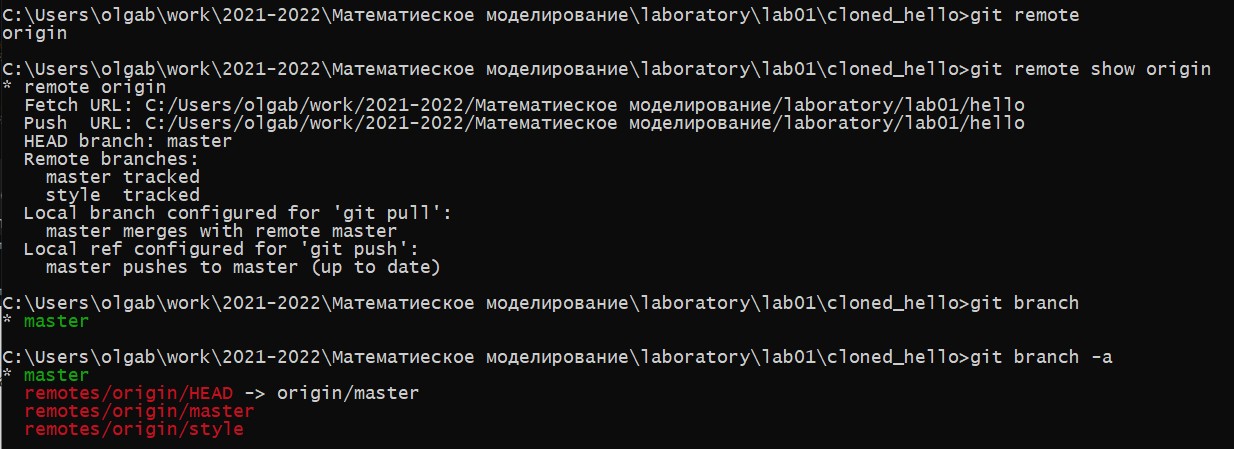
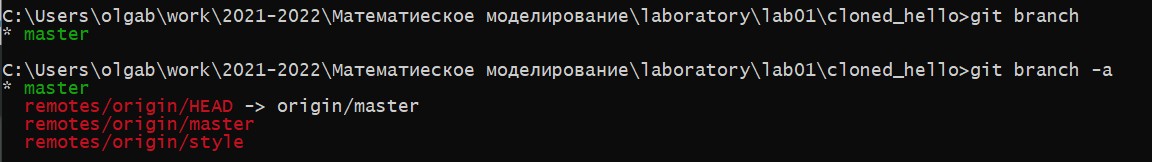
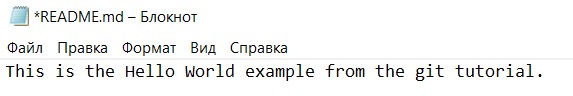
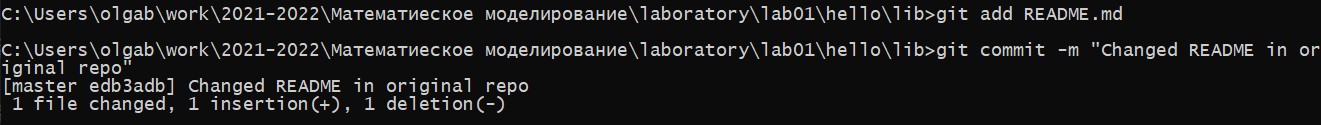
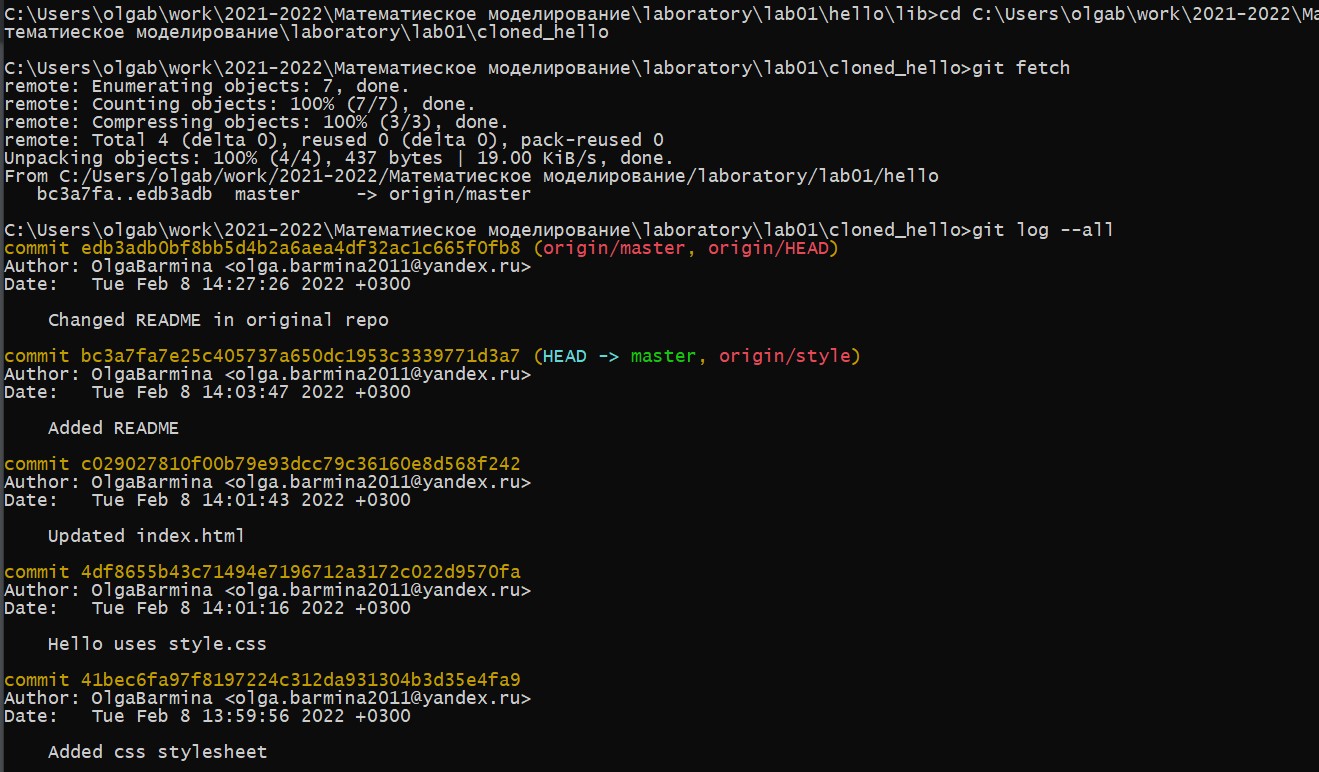
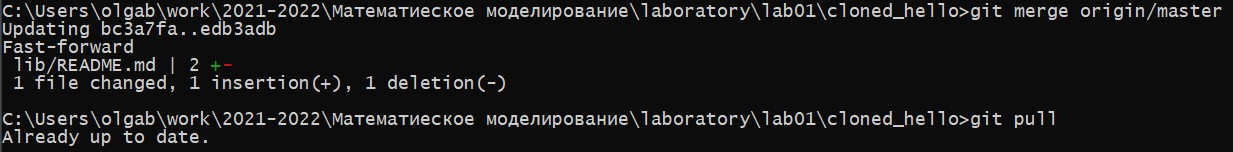
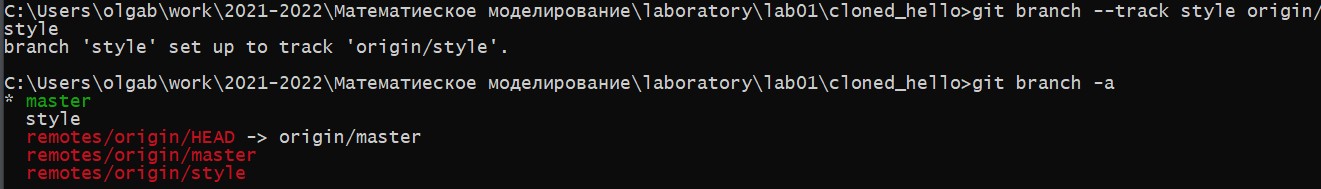
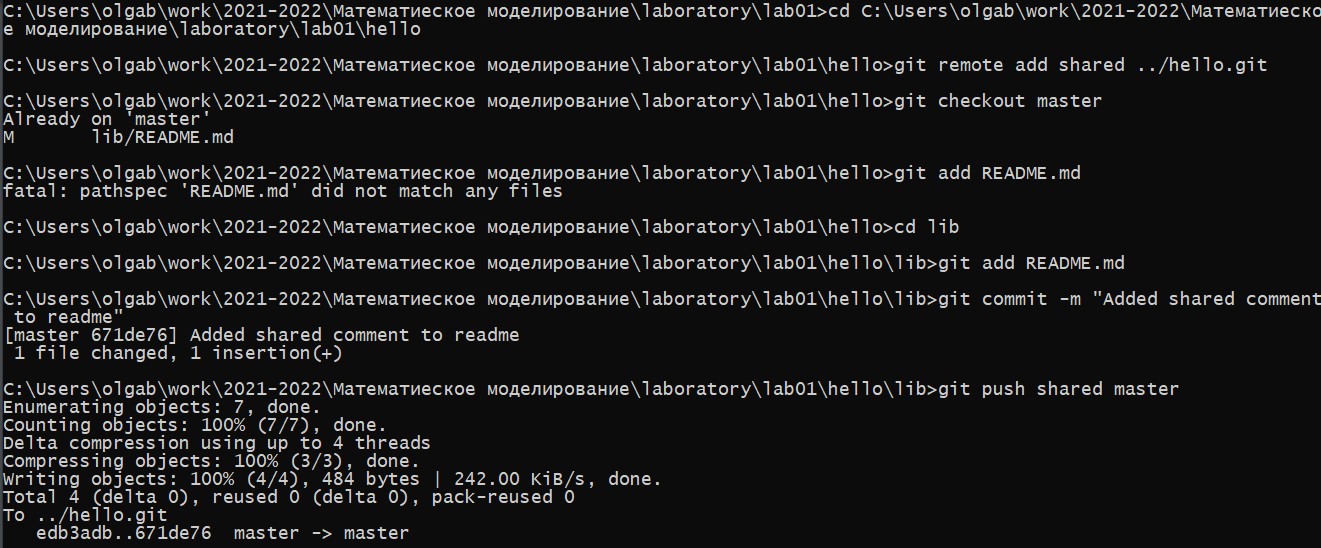
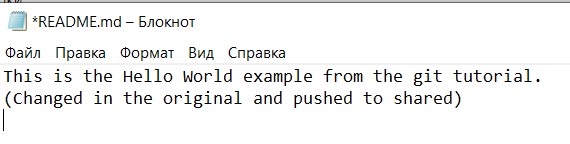
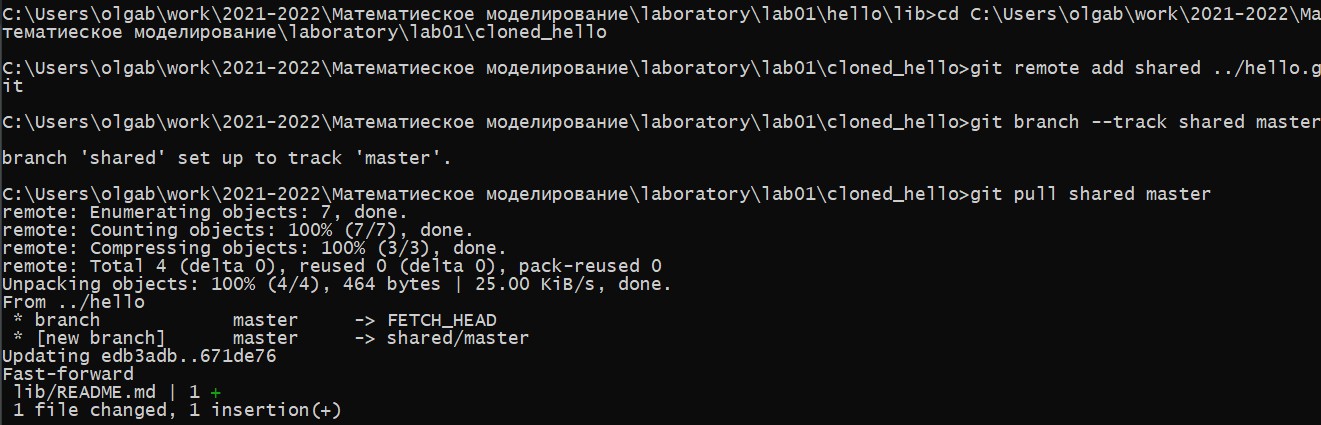
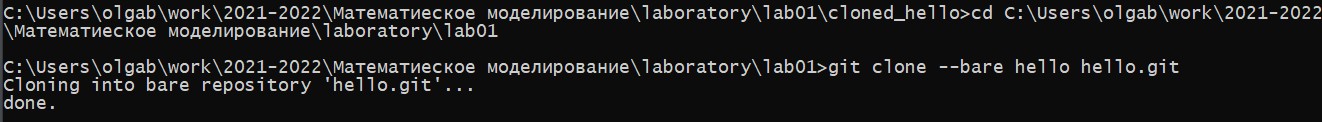
В ходе работы необходимо создать проект, научиться его изменять, индексировать и удалять изменения, создавать, изменять и отменять коммиты, перемещать файлы репозитория, работать с объектами git, создавать и перемещаться по ветками репозитория, совмещать ветки, клонировать репозитории, создавать чистые репозитории и добавлять удаленный репозиторий к другому репозиторию.

# Теоретическое введение

Описание некоторых команд файловой системы GNU Linux

|  |  |
| --- | --- |
| Имя команды | Описание команды |
| git config | Позволяет изменить стандартное поведение git |
| mkdir | Позволяет создать новый каталог |
| cd | Позволяет перейти в указанный каталог |
| echo | Позвляет записать информацию в файл прямо из консоли |
| git init | Создаёт git репозитории и директорию .git в текущей директории |
| git commit | Позволяет сделать коммит изменений |
| git status | Позволяет просмотреть состояния файлов в репозитории |
| git log | Позволяет просматривать историю коммитов репозитория |
| git checkout | Позволяет загружать ветку на новую ветку |
| git tag | Позволяет дать текущемму коммиту название |
| git reset | Позволяет сбросить HEAD текущего репозитория |
| git cat-file | Предоставляет информацию о содержимом, типе и размере для объектов репозитория |
| git merge | Сливает 2 ветки git репозитория |
| git rebase | Позволяет измененить коммитом в вашей истории |
| git clone | Клонирует репозиторий |
| git remote | Используется для работы с внешними репозиториями |
| git branch | Используется для создания ветки используйте |
| git pull | Используется для загрузки данных с внешнего репозитория |

# Выполнение лабораторной работы

1. Настраиваем core.autocrlf с параметром true для переводы строк текстовых файлов в главном репозитории в одинаковый формат. Задаем core.safecrlf как true. Устанавливаем флаг для преобразования Unicode символов в ASCII. 
2. Начинаем работу в пустом рабочем каталоге с создания пустого каталога с именем hello, затем входим в него и создаем там файл с именем hello.html.  Чтобы создать git репозиторий из этого каталога, выполняем команду git init. Добавим файл в репозиторий. 
3. Изменим содержимое файла hello.html. Сделаем коммит без метки -m.  
4. Изменим страницу «Hello, World», чтобы она содержала стандартные теги html и body. Добавляем это изменение в индекс git. Теперь добавляем заголовки HTML к странице «Hello, World». Проверяем текущий статус. Произведем коммит проиндексированного изменения, а затем еще раз проверим состояние. Теперь добавим второе изменение в индекс, а затем проверим состояние с помощью команды git status. Второе изменение было проиндексировано и готово к коммиту. Сделаем коммит второго изменения.    
5. Получим список произведенных изменений. Изучим данные лога и найдем хэш для первого коммита. Вернемся к последней версии в ветке master. Назовем текущую версию страницы hello первой (v1). Создадим тег для версии, которая идет перед текущей версией и назовем его v1-beta. Перейдем на основную ветку. Внесем изменение в файл hello.html в виде нежелательного комментария. Проверим состояние рабочего каталога. Используйте команду git checkout для переключения версии файла hello.html в репозитории.  рис 12  рис 14  
6. Внесите изменение в файл hello.html в виде нежелательного комментария. Проиндексируйте это изменение. Выполните сброс буферной зоны. Переключитесь на версию коммита. Далее мы отменим коммит путем создания нового коммита, отменяющего нежелательные изменения. Изменим файл hello.html. Сделаем коммит с новыми изменениями, отменяющими предыдущие. Проверим лог.     
7. Прежде чем удалить коммиты, давайте отметим последний коммит тегом oops. Глядя на историю лога, мы видим, что коммит с тегом «v1» является коммитом, предшествующим ошибочному коммиту. Сбросим ветку до этой точки. Посмотрим на все коммиты. Тег oops свою функцию выполнил. Удалим его и коммиты, на которые он ссылался.   
8. Добавим в страницу комментарий автора. После совершения коммита обновляем страницу hello, включив в нее email. Изменим предыдущий коммит, включив в него адрес электронной почты. Просмотрим лог.    
9. Сейчас мы собираемся создать структуру нашего репозитория. Давайте перенесем страницу в каталог lib. сделаем коммит этого перемещения: Добавим файл index.html в наш репозиторий. Добавьте файл и сделайте коммит.    
10. Произведем поиск последнего коммита. Выведем последний коммит с помощью SHA1 хэша. Выведем дерево каталогов используя SHA1 хэш из строки «дерева». Выведем каталог lib и файл hello.html. Исследуем git репозиторий вручную самостоятельно. Найдем оригинальный файл hello.html с самого первого коммита вручную по ссылкам SHA1 хэша в последнем коммите.   
11. Создадим новую ветку «style». Добавим файл стилей style.css. Обновим файлы hello.html и index.html, чтобы использовать стили style.css.      
12. Просмотрим полный лог. Переключимся на ветку master. Создадим файл README в ветке master. Сделаем коммит изменений README.md в ветку master. Просмотрим текущие ветки. Вернемся к ветке style и сольем master с style.    
13. Вернемся в ветку master и внесем изменения в файл lib/hello.html. Закоммитим изменения. Просмотрим ветки.   рис 47 
14. Вернемся к ветке style и попытаемся объединить ее с новой веткой master. Вручную внесем изменения в lib/hello.html. Сделаем коммит решения конфликта.   
15. Вернемся на ветке style к точке перед тем, как мы слили ее с веткой master. Видим, что коммит «Updated index.html» был последним на ветке style перед слиянием. Сбросим ветку style к этому коммиту. Вернемся в ветке master в точку перед внесением конфликтующих изменений. Коммит «Added README» идет непосредственно перед коммитом конфликтующего интерактивного режима. Сбросим ветку master к коммиту «Added README».    
16. Используем команду rebase вместо команды merge. Сольем изменения style в ветку master.  
17. Перейдем в рабочий каталог. Создадим клон репозитория hello. Взглянем на клонированный репозиторий. Просмотрим origin клонированого репозитория. Посмотрим на ветки, доступные в клонированном репозитории. рис 58    
18. Внесем изменения в оригинальный файл README.md репозитория hello. Добавим это изменение и сделаем коммит. Научимся извлекать изменения из удаленного репозитория с помощью git fetch. Сольем извлеченные изменения в локальную ветку master. Попробуем ввесли команду git pull, которая совмещаем в себе fetch и merge. Добавим локальную ветку, которая отслеживает удаленную ветку.     
19. Создадим чистый репозиторий. Добавим репозиторий hello.git к нашему оригинальному репозиторию. Отправим изменения в другие репозитории. Для этого отредактируем файл README.md и сделаем коммит. Научимся извлекать изменения из общего репозитория. Переключимся в клонированный репозиторий и извлечем изменения, только что отправленные в общий репозиторий.    

# Выводы

В ходе работы мы изучили основы работы с git репозиториями. Мы научились создавать и изменять проект, индексировать и удалять изменения, создавать, изменять и отменять коммиты, перемещать файлы репозитория, работать с объектами git, создавать и перемещаться по ветками репозитория, совмещать ветки, клонировать репозитории, создавать чистые репозитории и добавлять удаленный репозиторий.

# Список литературы