**Отчёт о лабораторной работе №3.**

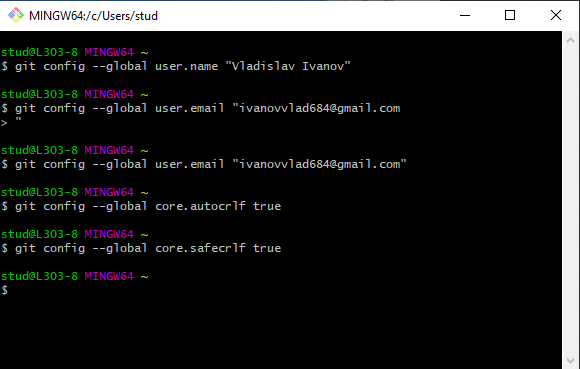
**Отчёт выполнил Иванов Владислав Григорьевич**

**Группа БПИ21-01 02**

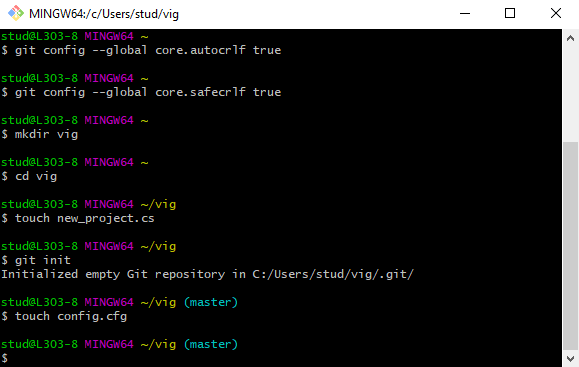
**Ход работы**

**7 Вариант:** Использовать в работе 2 разных файла, для одного из них использовать 3 различные ветки.

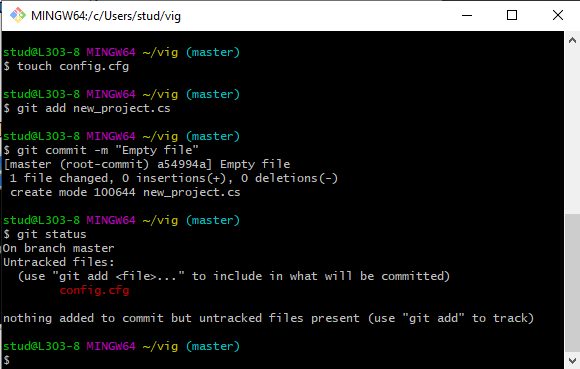
1. Первоначальная настройка Git

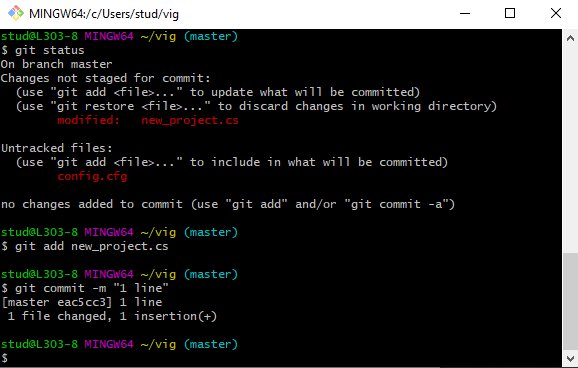
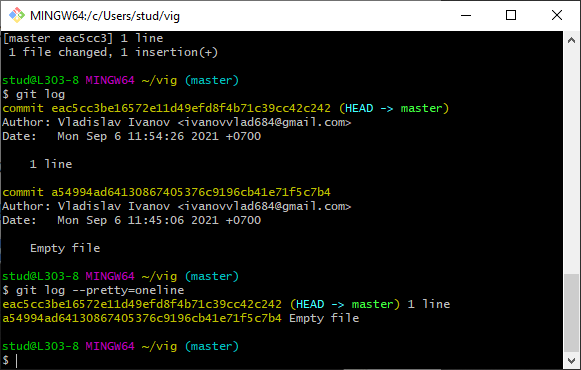


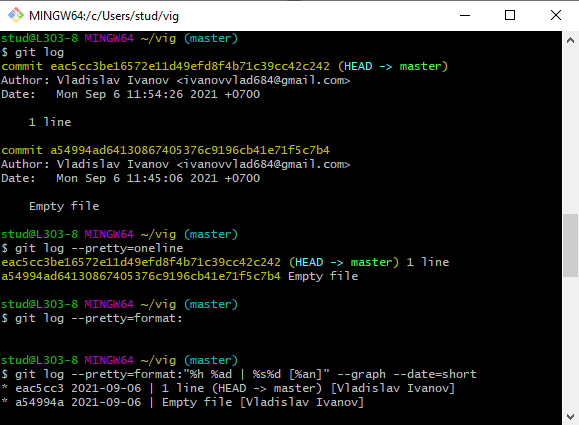
1. Создание директории vig. Создание двух новых файлов new\_project.cs и config.cfg, а так же инициализация нового репозитория (git init)



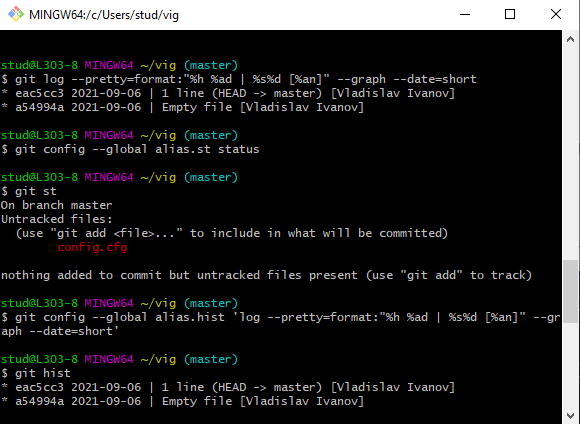
1. Добавление коммита для файла new\_project.cs и проверка состояния репозитория (коммит для config.cfg ещё не был добавлен).



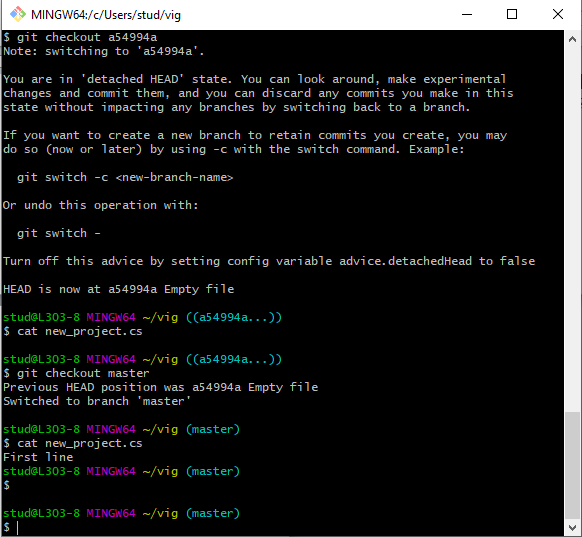
1. Проверка статуса файла new\_project.cs после внесения в него изменений. Добавление нового коммита.
2. Просмотр истории всех версий файла new\_project.cs. Вывод истории в режиме “одной строки”.
3. Вывод форматированной истории



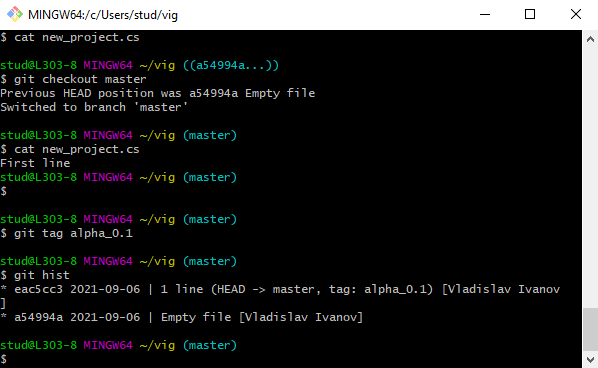
1. Добавление двух алиасов, которые отображают статус и форматированную историю



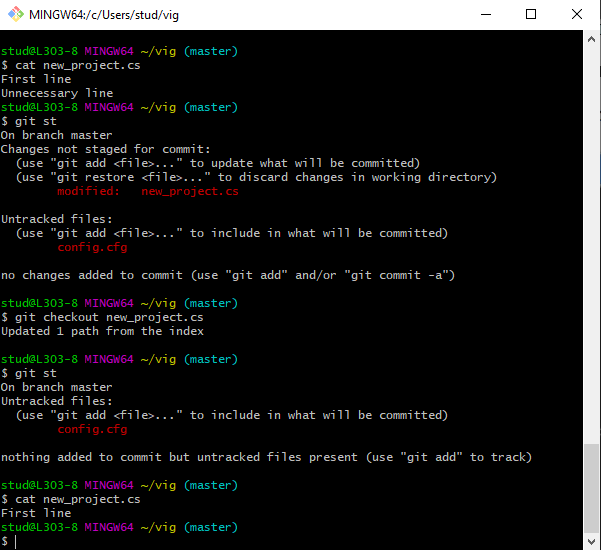
1. Возвращение к предыдущей версии файла с коммитом “Empty”. Команда $cat демонстрирует, что теперь файл new\_project.cs пуст. При помощи команды $git checkout master мы возвращаемся к последней версии файла. Команда $cat демонстрирует, что теперь файл new\_project.cs имеет в себе одну строчку.

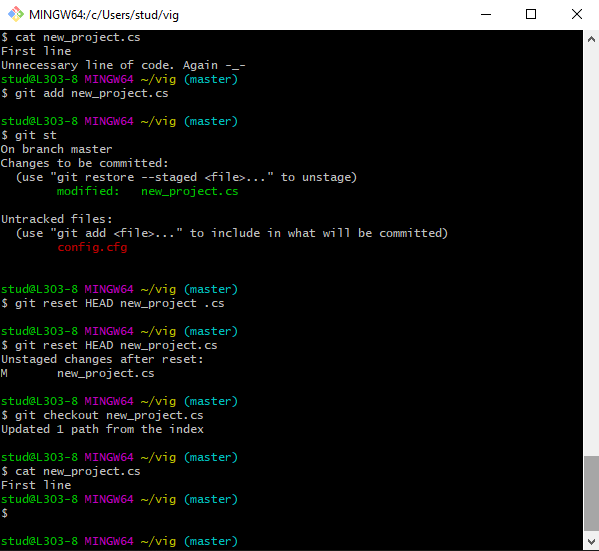
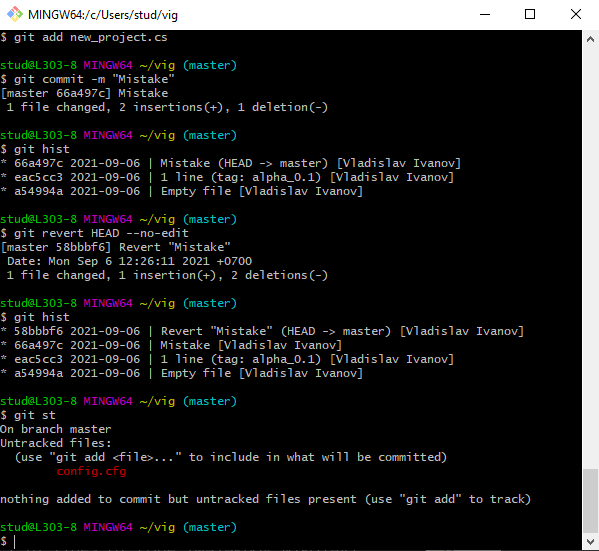
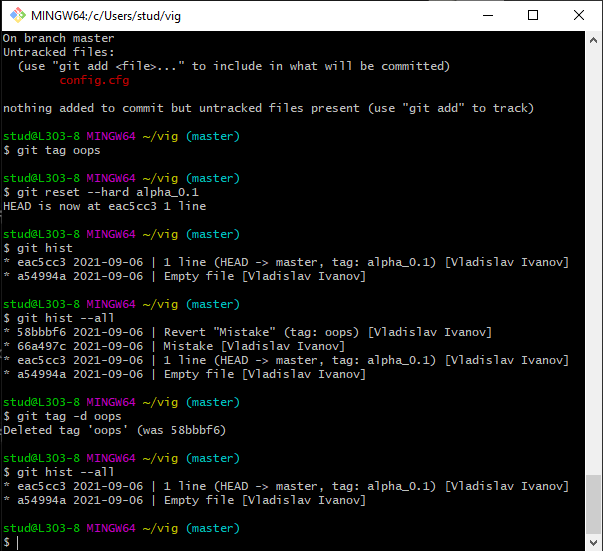
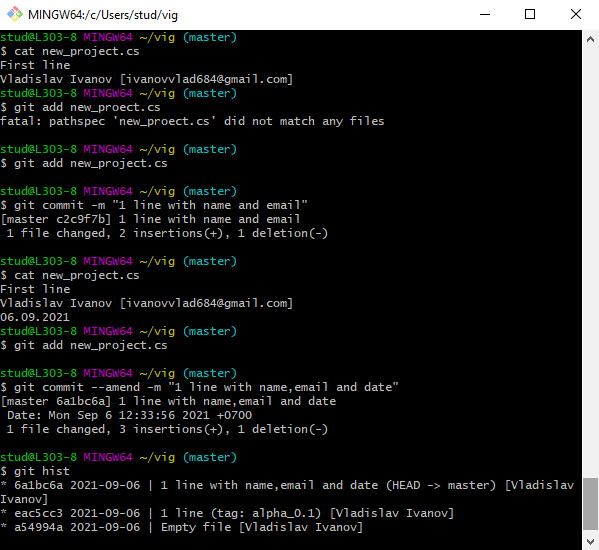
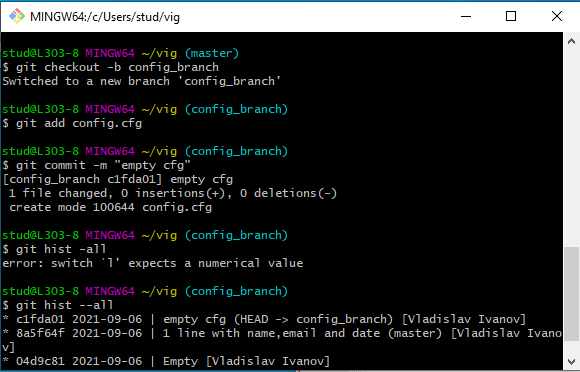
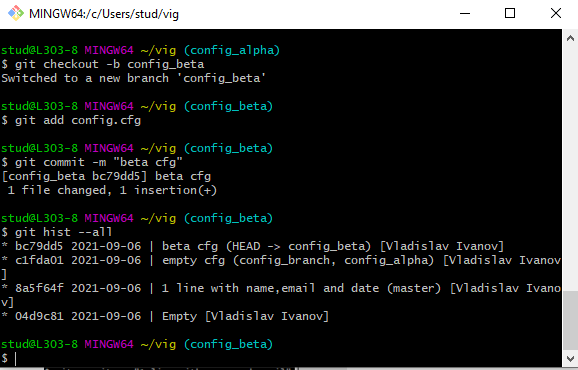


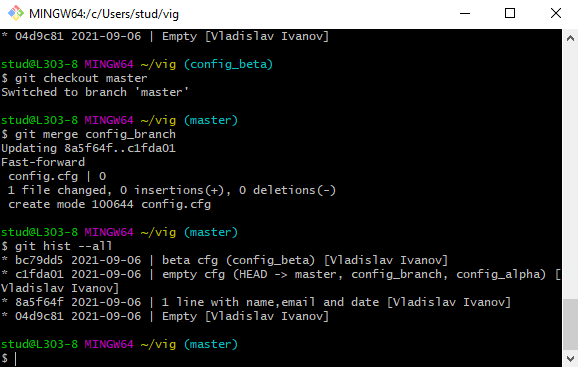
1. Тегирование текущей версии файла. Демонстрация тега в истории версий.



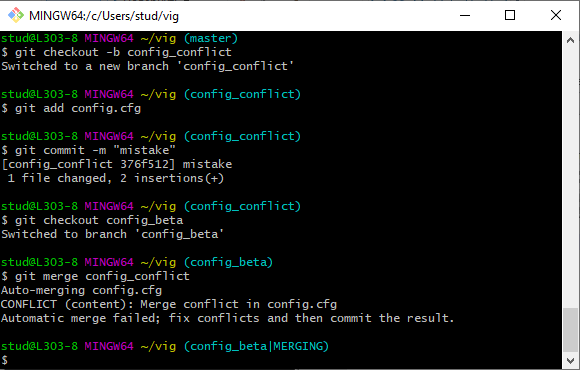
1. В файл была добавлена ненужная строка. При помощи команды get checkout new\_project.cs можно вернуться к последней версии файла.

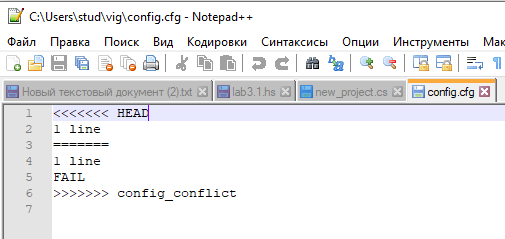


1. В файл снова была добавлена ненужная строка кода, но в этот раз он был проиндексирован. При помощи команд git reset HEAD new\_project.cs и git checkout new\_project.cs можно откатить изменения.
2. Файл new\_project.cs снова обзавёлся ненужной строчкой кода. Он был случайно проиндексирован и к нему добавился коммит. В таком случае при помощи команды git revert HEAD –no-edit можно откатить к последней версии, но данный коммит всё равно останется в истории файлов
3. Для удаления ошибочной версии из истории необходимо добавить к ней тег (oops). После этого надо откатиться к последней версии файла и удалить тег oops.
4. Можно исправить уже проиндексированный коммит без его удаления при помощи команды git commit –amend m.
5. Создание ветки config\_branch и добавление в неё файла config.cfg 
6. Создание ещё одной ветки config\_beta и добавление в неё изменённого файла config.cfg
7. Слияние веток master и config\_branch

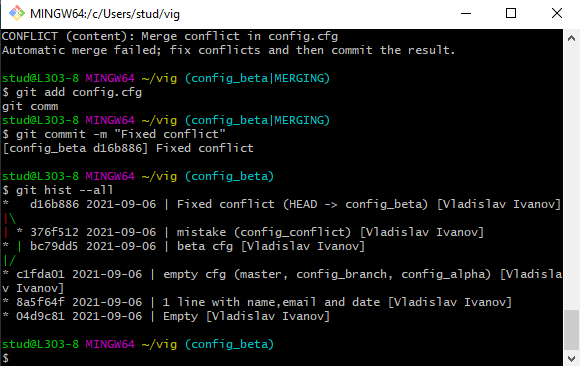


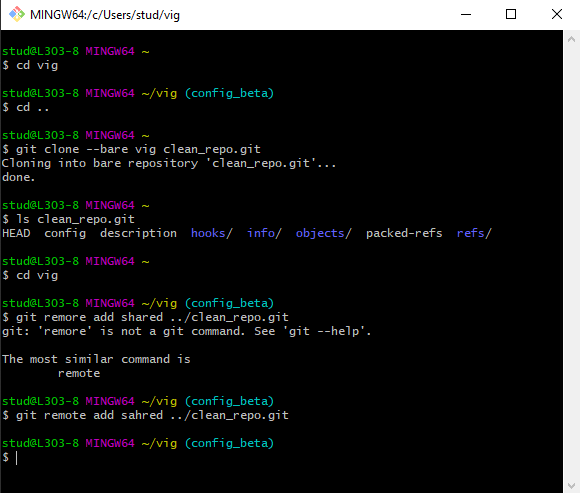
19.Создание конфликтной ветки config\_conflict. В эту ветку был добавлен файл config.cfg с изменениями. Далее данная ветка была объединена с веткой config\_beta, где файл config.cfg имел иное содержание. В итоге, случился конфликт:





20. После изменения файла вручную, во вторую ветку был добавлен испралвенный файл config.cfg с коммитом “Fixed conflict”

.

21. Создание чистого репозитория clean\_repo

**Вывод:** Используя систему управления версиями Git, я научился создавать репозиторий, в котором можно хранить разные версии файлов и отслеживать историю их создания и изменений. При помощи алиасов я упростил работу с Git, превратив большие команды в короткие. Также я научился создавать теги для определённых версий файла и отменять ненужные коммиты. Более того было освоено создание веток и решение конфликтов при их объединении. Система управления версиями Git очень удобна из-за своего функционала и архитектуры.

**Ответы на контрольные вопросы**

1. Git – это ПО, позволяющее отслеживать изменения файлов, просматривать, кто и когда внёс измениея, при необходимости совершать откат.
2. Git предоставляет возможности по отслеживанию изменений в проекте (кто и когда внёс изменения), а также позволяет создавать новые и объединять уже существующие ветви проекта, более того Git производит контроль доступа пользователей к проекту.
3. Коммит – это действие, с помощью которого можно отправить изменённый файл в репозиторий.
4. Историю коммитов можно вывести на экран при помощи команды $git log.
5. Для создания пустого репозитория в директории необходимо прописать команду git init.
6. Репозиторий состоит из места, где производится работа с фалами (Working copy), и репозитория, куда отправляются файлы (repository). Чтобы отправить файл в репозиторий необходимо создать commit. Чтобы забрать файл из репозитория, нужно сделать pull.
7. Создать новую ветку можно при помощи команды $git checkout –b <branch name>.Для объединения веток используется команда $git merge <branch name>.
8. Staging area – это область подготовленных файлов. Данная область содержит в себе данные о том, что должно войти в следующий коммит.
9. Сравнить версии файла можно при помощи команды $git diff.