**Лабораторная работа 3**

**Вариант 5: Использовать в работе 2 разных файла, и количество версий каждого файла не должно быть меньше 5.**

В ходе выполнения работы необходимо ознакомиться с системой контроля версий GIT и выполнить в ней основные виды работ.

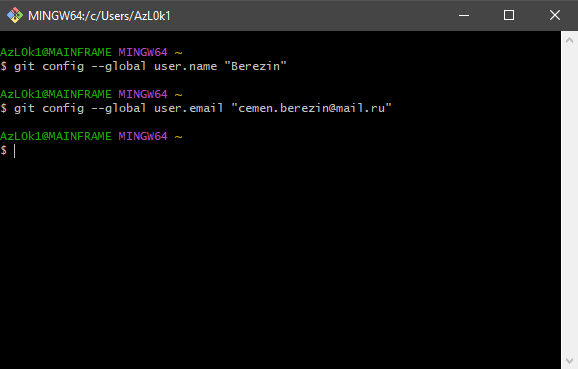


Рисунок 1 – Настройка имени и электронной почты

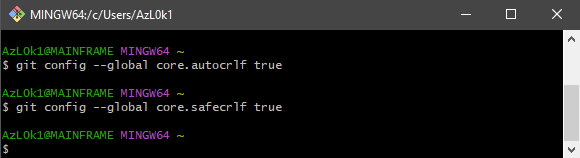


Рисунок 2 – Настройка параметров окончания строк

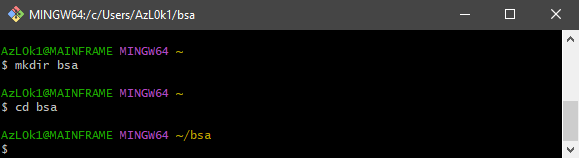


Рисунок 3 – Создание каталога



Рисунок 4 – Создание файла

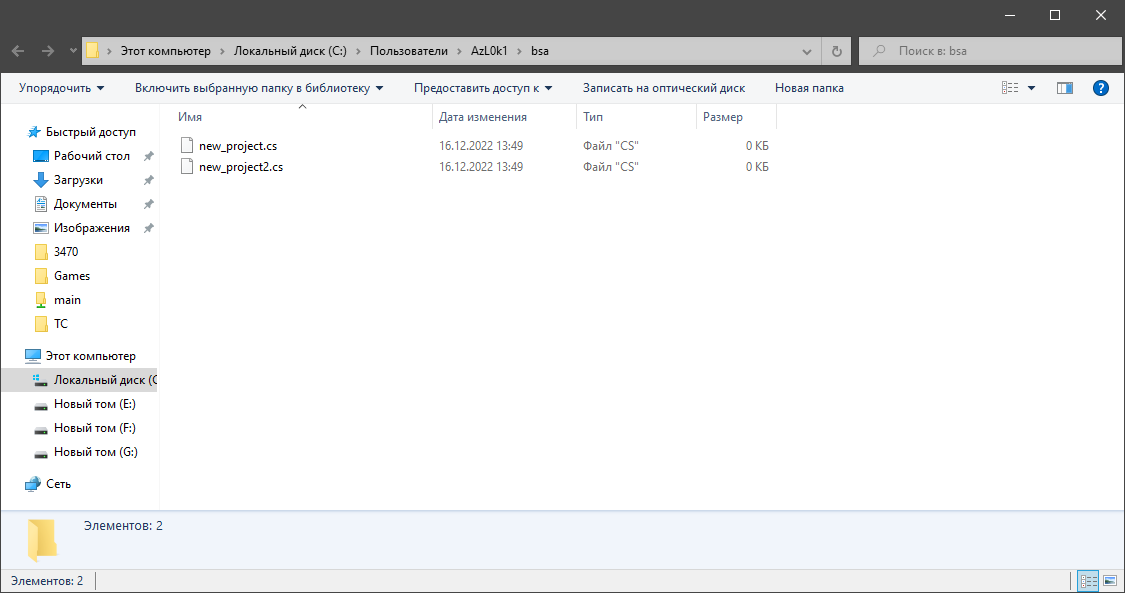


Рисунок 5 – созданный файл

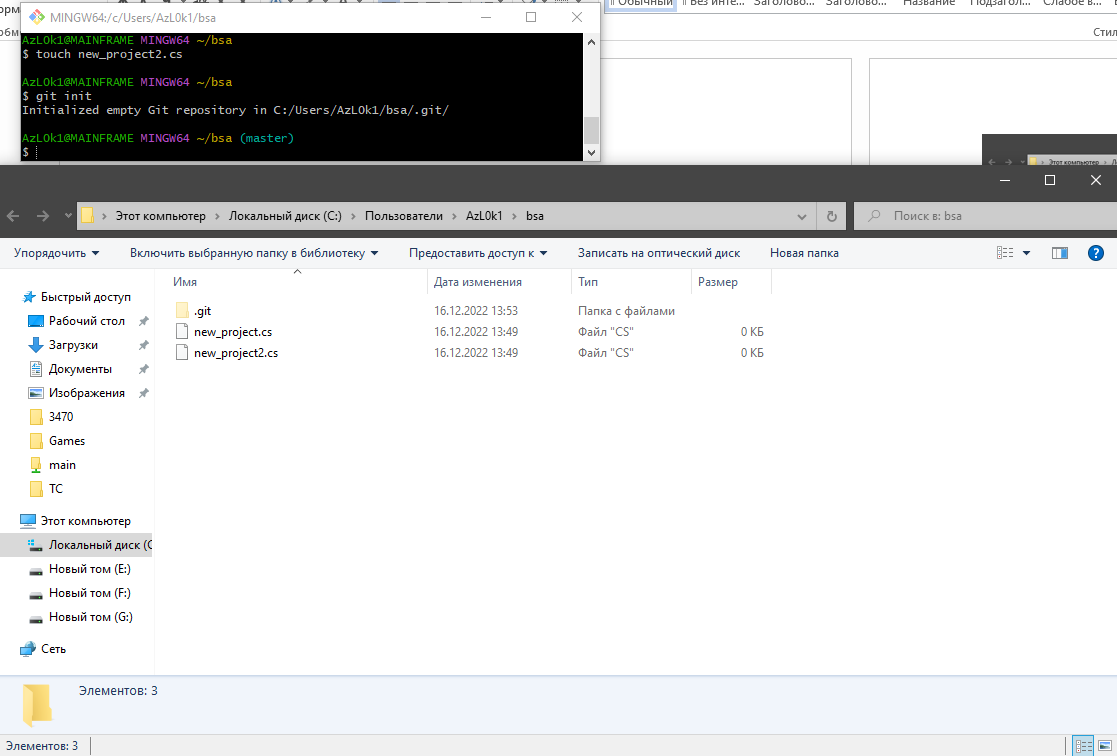


Рисунок 6 – Результат создания репозитория

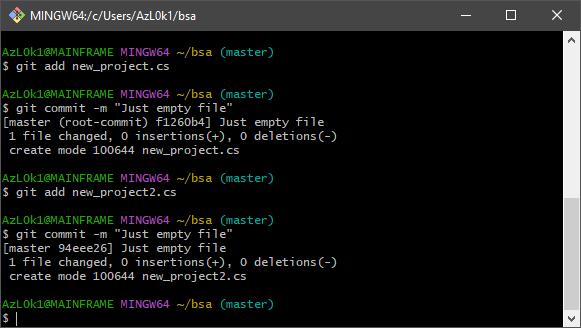


Рисунок 7 – Результат добавления первого коммита

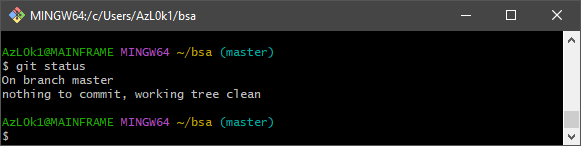


Рисунок 8 – Проверка состояния репозитория

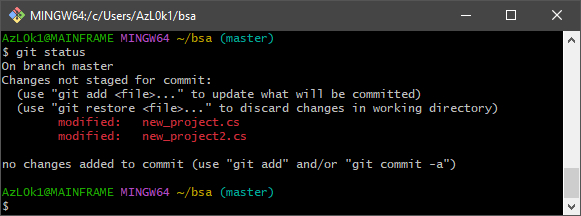


Рисунок 9 – Результат проверки состояния репозитория, после изменения исходного файла

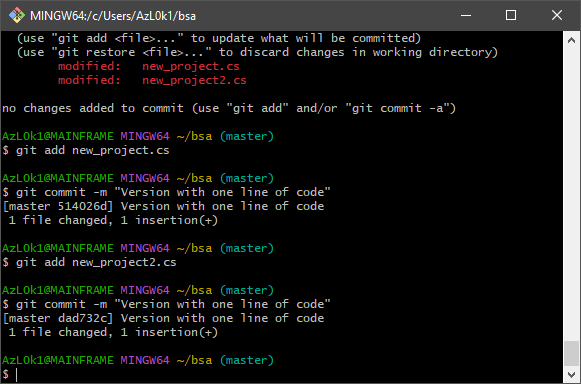


Рисунок 10 – Проверка состояния репозитория и добавление коммита

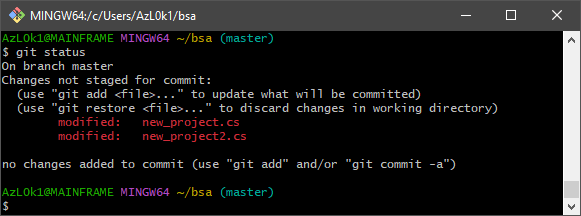


Рисунок 11 – Очередная проверка состояния репозитория

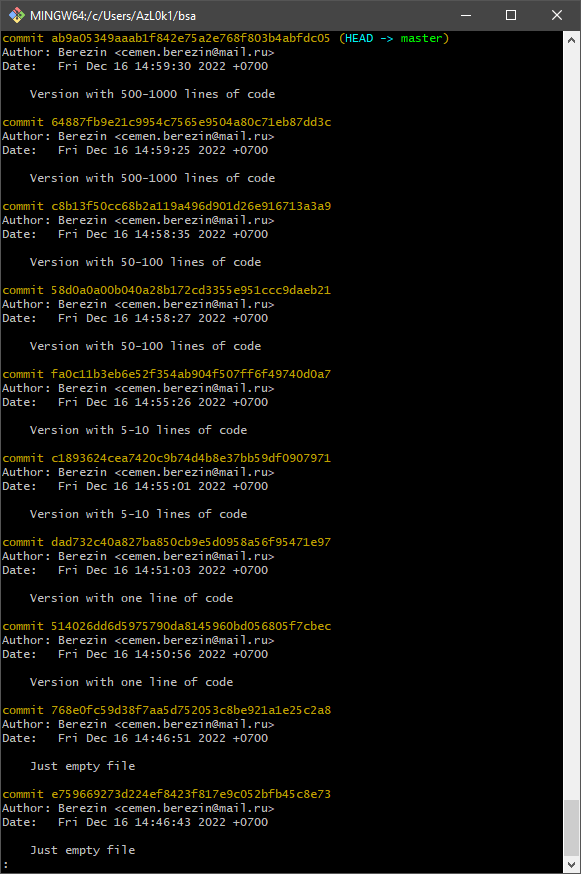


Рисунок 12 – История проекта

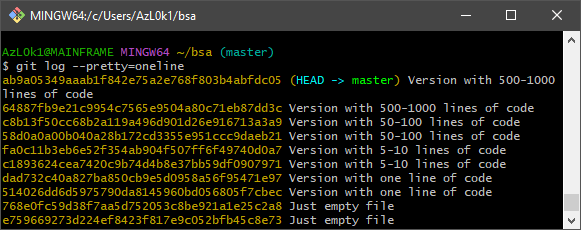


Рисунок 13 – История проекта в режиме «одной строки»

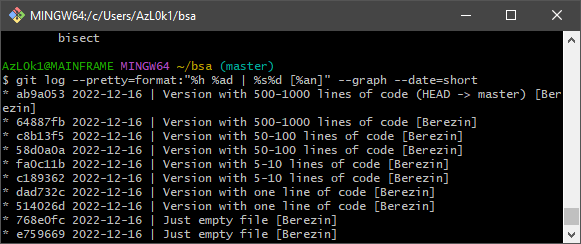


Рисунок 14 – Пример форматированной истории

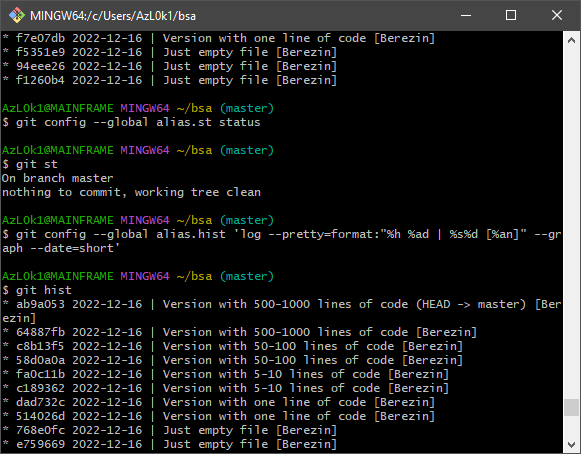


Рисунок 15 – Использование алиасов

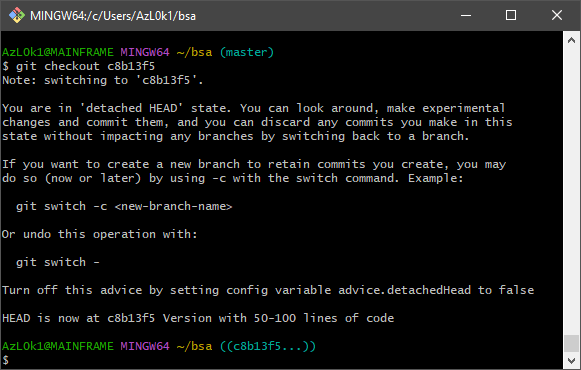


Рисунок 16 – Возврат к прошлым версиям

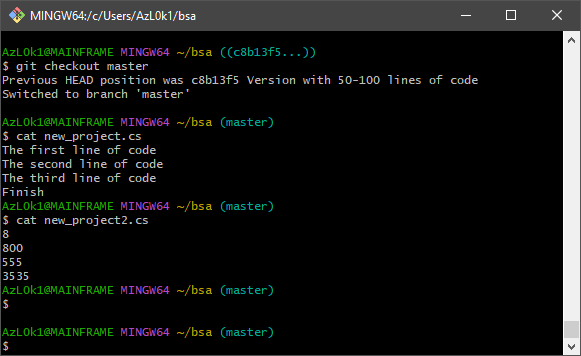


Рисунок 17 – Пример возврата к последней версии файла

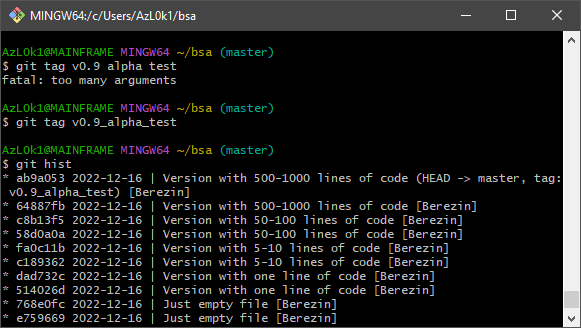


Рисунок 18 – Пример использования тегов в отслеживании версий

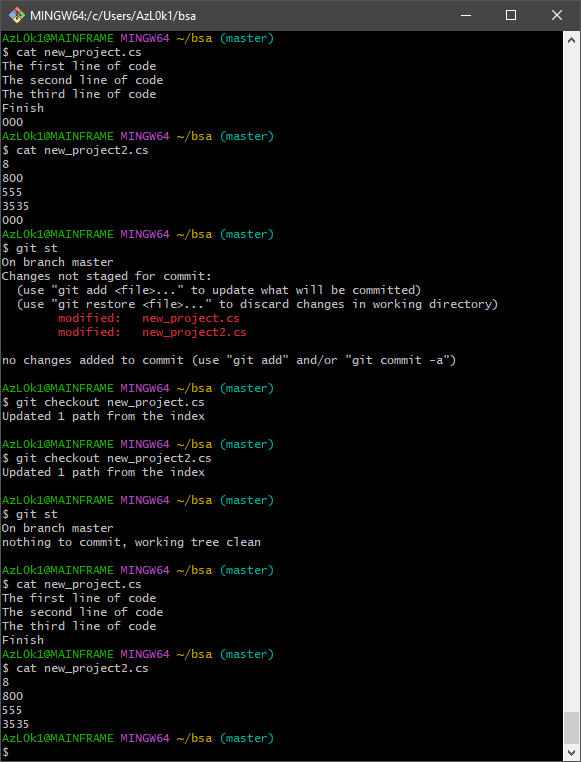


Рисунок 19 – Пример отмены изменений

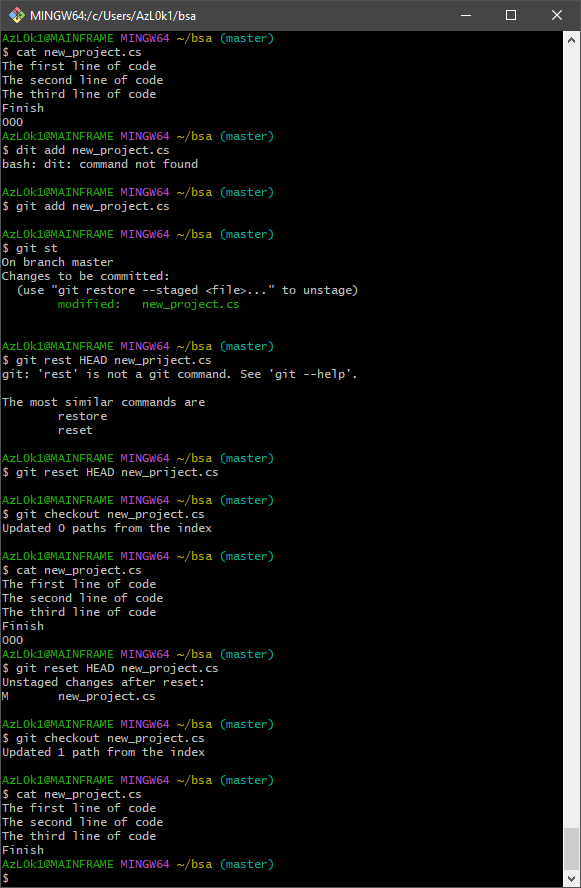


Рисунок 20 – Пример отмены проиндексированных изменений

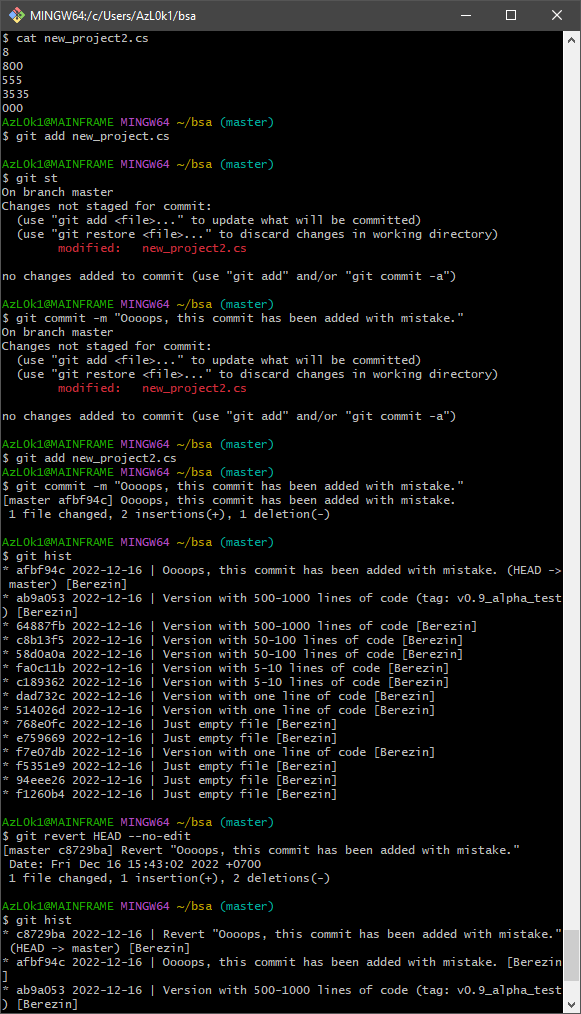


Рисунок 21 – Пример отмены неправильного коммита

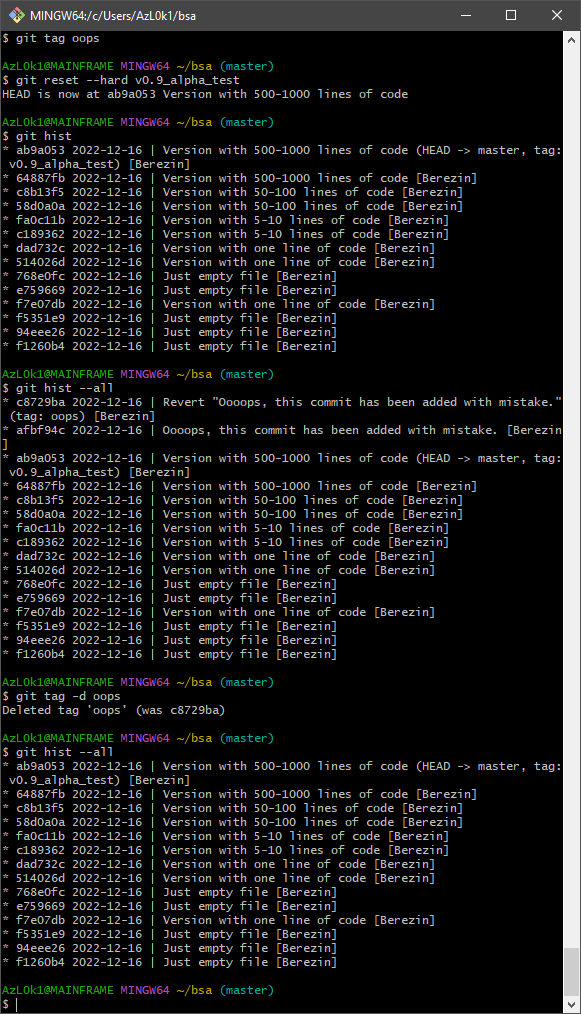


Рисунок 22 – Пример удаления

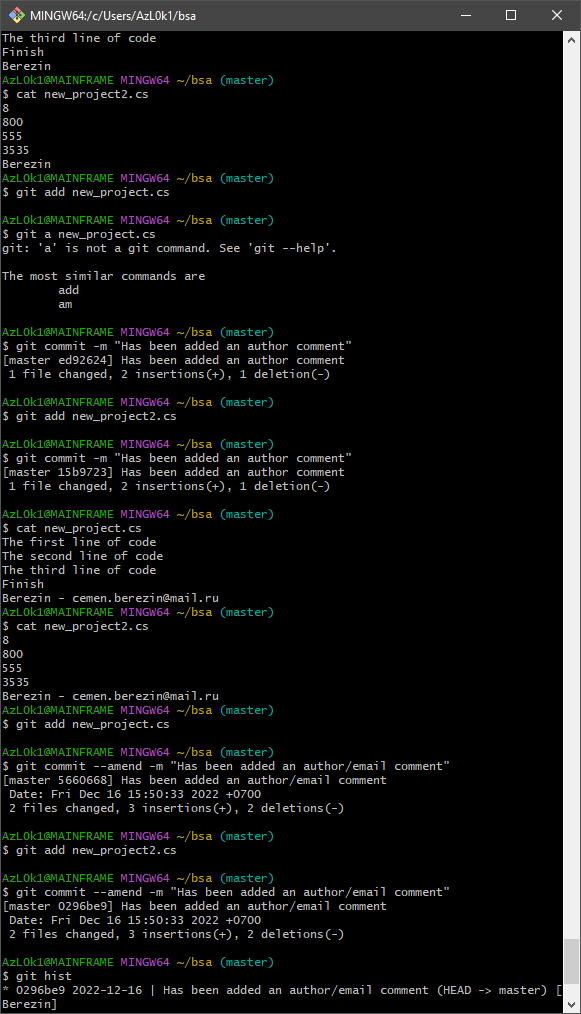


Рисунок 23 – Исправление коммита

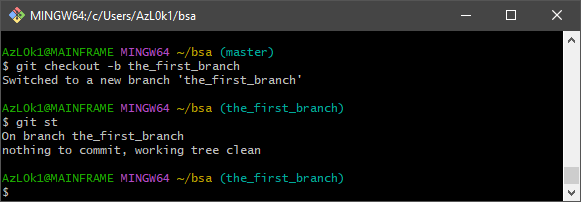


Рисунок 24 – Создание ветки the\_first\_branch

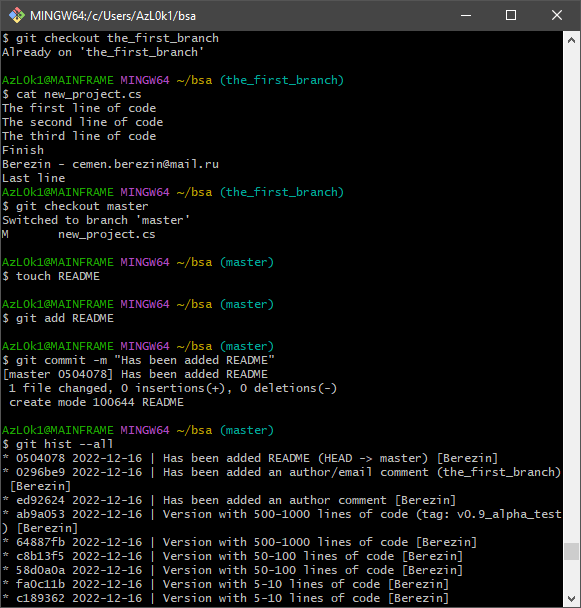


Рисунок 25 – Работа с ветками

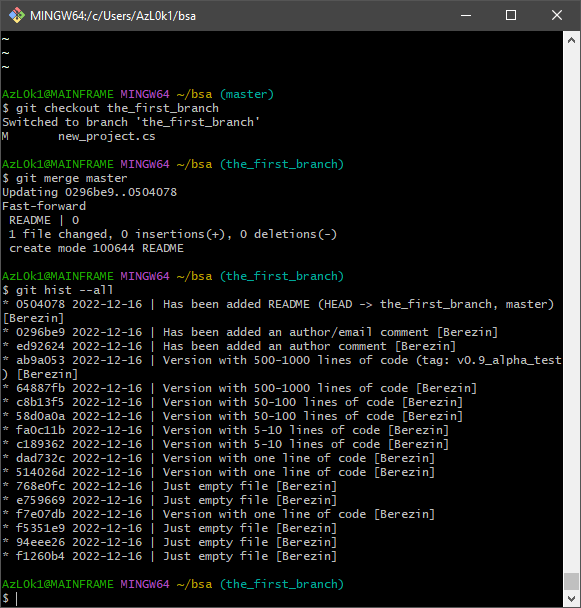


Рисунок 26 – Слияние веток

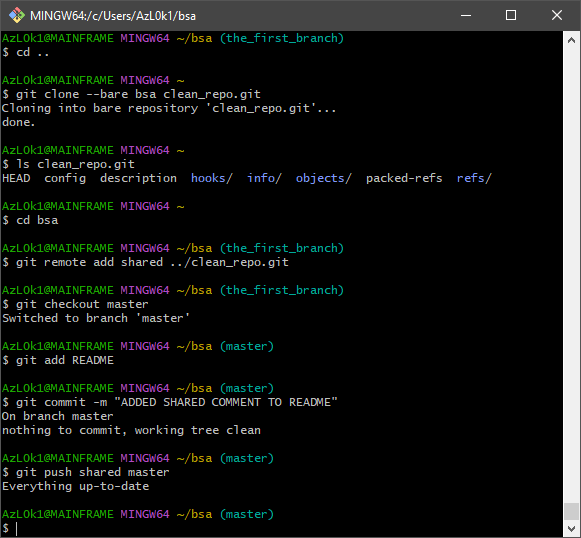


Рисунок 27 – Создание репозитория clean\_repo

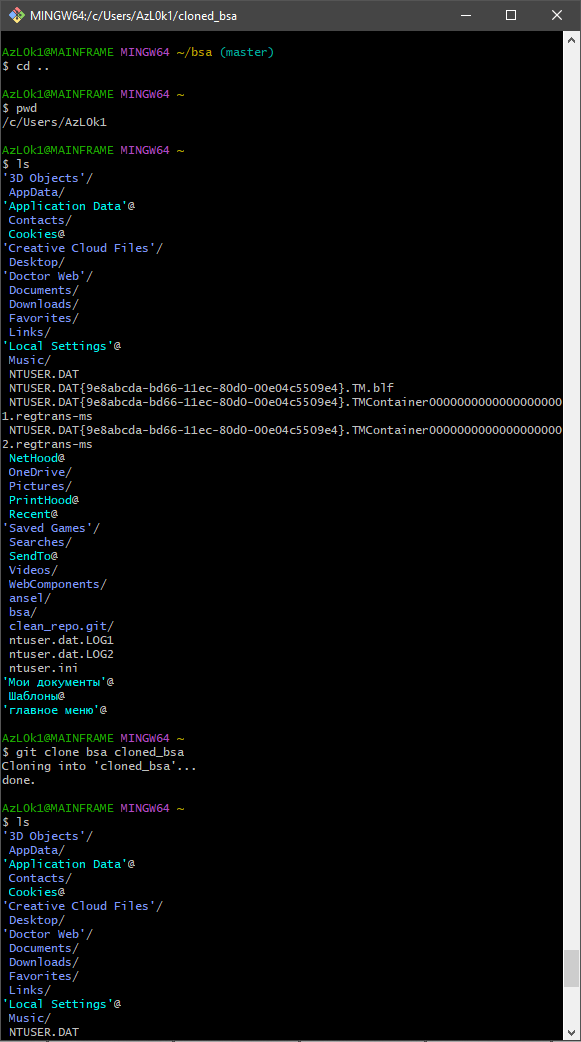


Рисунок 28 – Работа с удаленным репозиторием (часть 1)

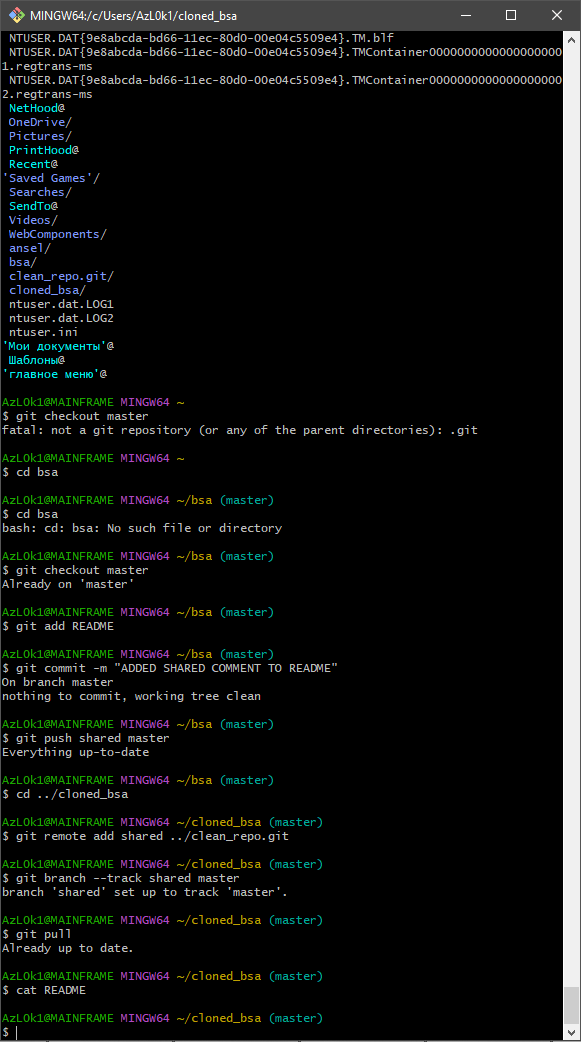


Рисунок 29 – Работа с удаленным репозиторием (часть 2)

В ходе выполнения работы удалось выполнить основные типы работ и тем самым ознакомиться с работой системы контроля версий GIT.

**Контрольные вопросы**

1. Что представляет собой система GIT?

- GIT – это система контроля версий, позволяющая отслеживать изменения в документах, производить откат и определять кто и когда внес изменения.

2. Перечислите возможности систем управления версиями.

- Система управления версиями дает возможность видеть, кто, когда и какие изменения вносил; позволяет формировать новые ветви проекта, объединять уже имеющиеся; настраивать контроль доступа к проекту; осуществлять откат до предыдущих версий.

3. Что такое коммит и когда он выполняется?

- Коммит – это операция отправки изменений. Он выполняется, когда необходимо добавить файл или его изменения в репозиторий.

4. Как посмотреть историю коммитов?

- Историю коммитов можно посмотреть командой git log

5. Как создать пустой GIT репозиторий?

- Пустой репозиторий создаётся командой git init

6. Какую архитектуру имеет система контроля версий GIT?

- Система контроля версий *git*использует архитектуру трех деревьев, регламент работы которой выглядит так:

1. Перед началом работы разработчик делает *checkout*, для того чтобы быть уверенным, что он будет работать с актуальной рабочей копией.
2. Разработчик вносит необходимые изменения в исходный код.
3. Разработчик отправляет необходимый набор файлов, изменения в которые внесены, в *stage*, для того, чтобы потом построить из них коммит. До того, как изменения будут отправлены в репозиторий, разработчик может добавлять и удалять файлы из *stage*. Набор файлов в *stage*, как правило, идеологически связан между собой.
4. Разработчик отправляет изменения в репозиторий (коммитит их)
5. Пункты 2-4 повторяются необходимое количество раз

7. Как создать новую ветку в GIT и соединить ветки?

- Ветка создаётся командой git checkout –b <имя\_ветки>. Объединяются ветки командой git merge <имя\_ветки>

8. Поясните понятие «staging area».

- Staging area – это область подготовленных файлов, которая содержит информацию о том что должно войти в следующий коммит. Разработчик отправляет в stage необходимый набор файлов, в которые внесены изменения, чтобы потом построить из них коммит. До того, как изменения будут отправлены в репозиторий, разработчик может добавлять и удалять файлы из stage.

9. Как сравнить версии файла в GIT?

- Сравнить версии файла можно командой git diff