Отчёт по лабораторной работе №3

Вариант 10

(Использовать 3 ветки, создать 2 конфликта и разрешить их.)

Работу выполнил: Кокшаров Кирилл Сергеевич БПИ21-01

Ход работы:

1. С помощью команд

git config --global user.name "Your Name"

git config --global user.email your\_email@whatever.com

указываю своё имя и свой адрес электронной почты, а с помощью команд

git config --global core.autocrlf true

git config --global core.safecrlf true

устанавливаю параметры окончания строк, далее создаю и перехожу в каталог kks командами

mkdir «your initials»

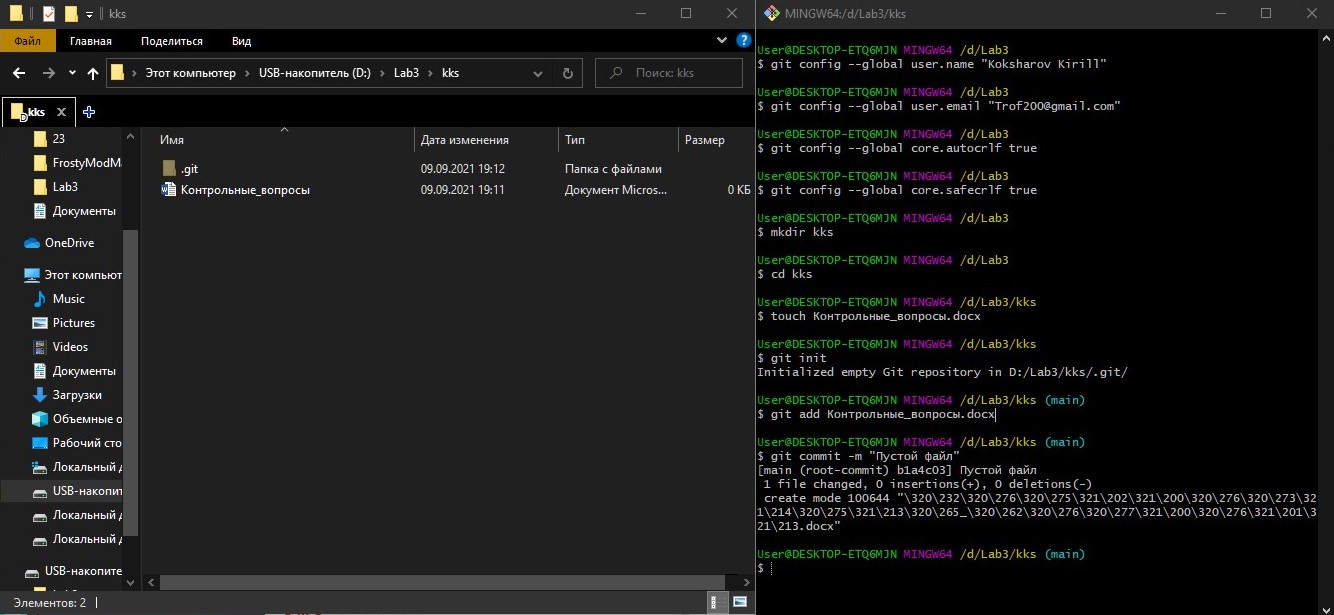
cd «your initials»

командой touch Контрольные\_вопросы.docx создаю новый файл. Командой

git init инициализирую новый репозиторий в папке kks. После этого добавляю файл Контрольные\_вопросы.docx в созданный репозиторий с помощью команд

git add Контрольные\_вопросы.docx

git commit –m “Пустой файл”



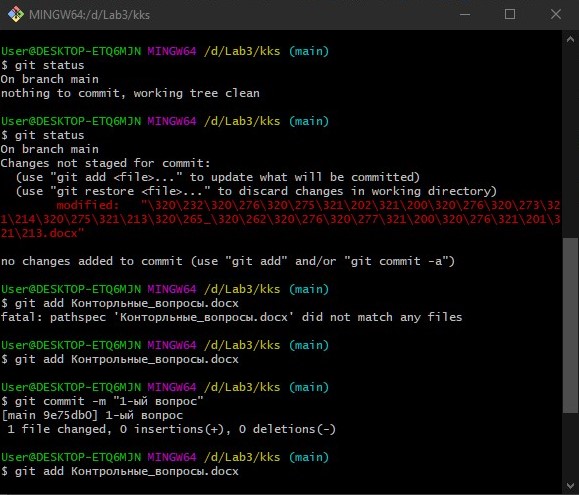
1. Командой

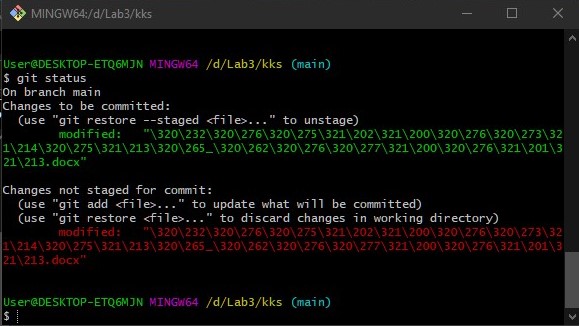
git status

проверяю состояние репозитория. После внесения изменений в файл, повторно проверяю состояние репозитория и замечаю, что его состояние изменилось. Затем идексирую изменения и добавляю коммит с помощью команд

git add Контрольные\_вопросы.docx

git commit –m “1-ый вопрос”

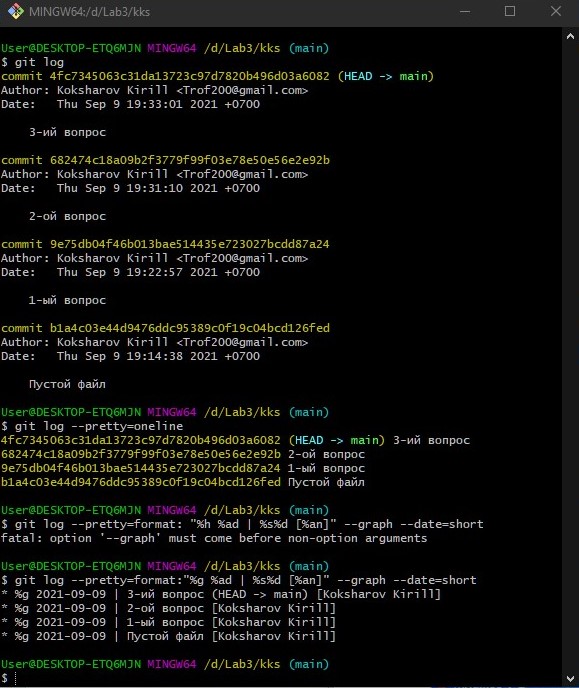


1. Далее я вношу изменения в файл, индексирую изменения, проверяю состояние и снова вношу изменения. После проверки состояния, замечаю, что система различает версии одного и того же файла, одна из которых проиндексирована и готова к добавлению коммита, а вторая не проиндексирована.
2. Для проверки истории версий использую команду

git log.

Для более удобного просмотра истории, использую режим просмотра в одну строчку, а затем, для повышения удобства, применяю следующее форматирование:

git log --pretty format:”%h %ad | %s%d [%an]” --graph --date=short



1. Используя команды

git config --global alias.st status

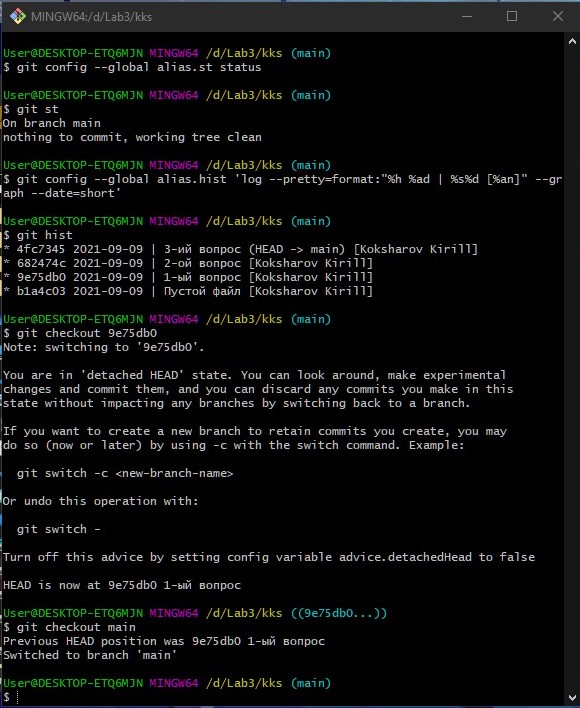
git config --global alias.hist ‘log --pretty=format:”%h %ad | %s%d [%an]” --graph --date=short’

создаю алиасы (сокрашения) для команд status и log.

После этого переключаюсь между версиями своего проекта, используя следующие команды:

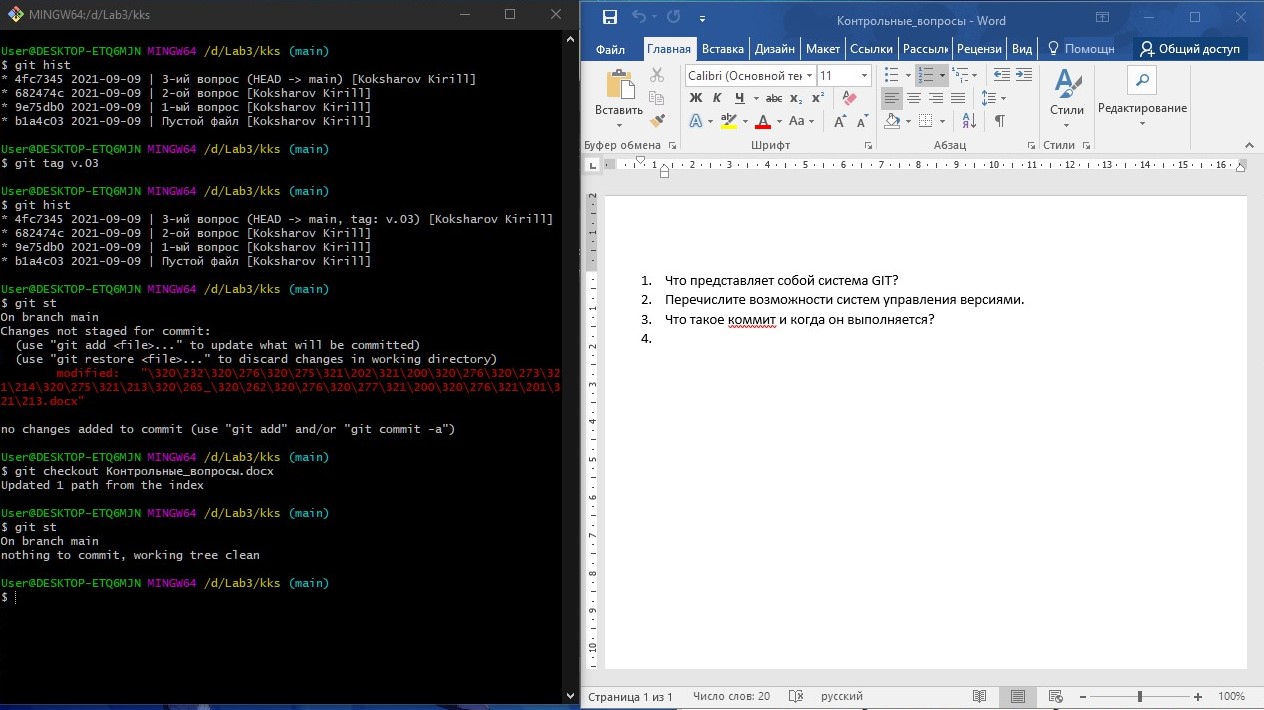
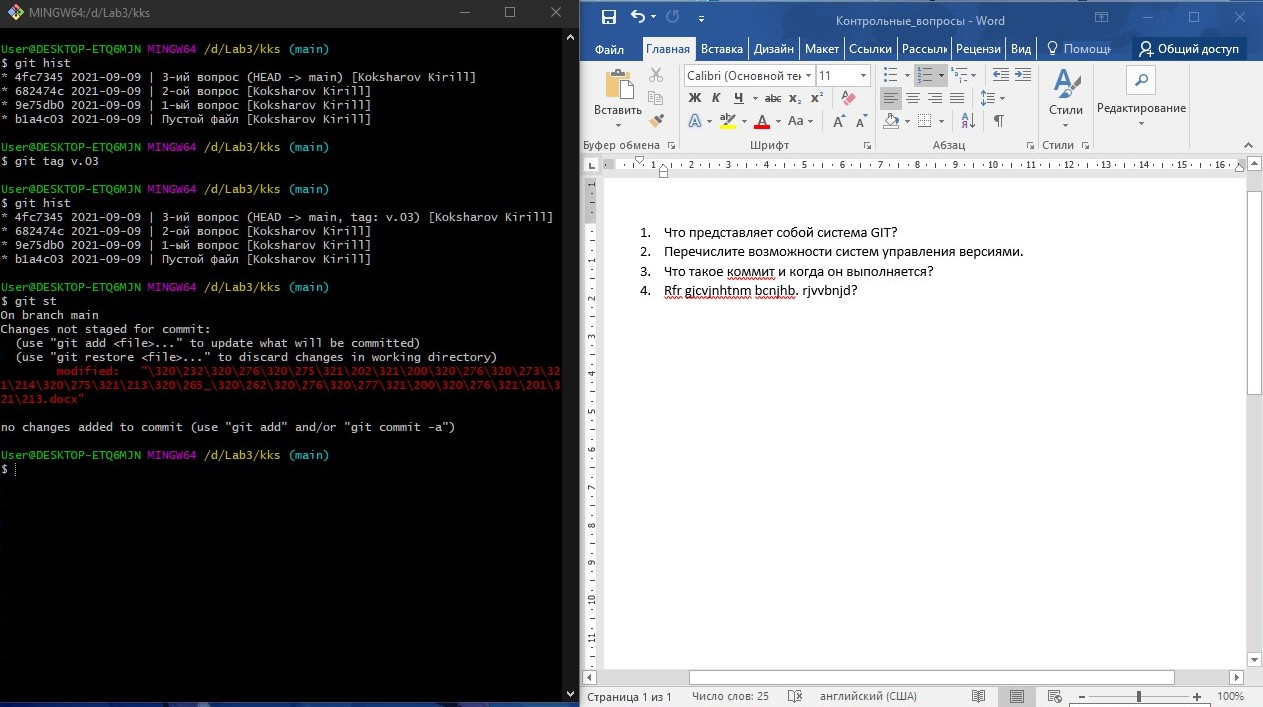
git checkout 9e75db0

git checkout main

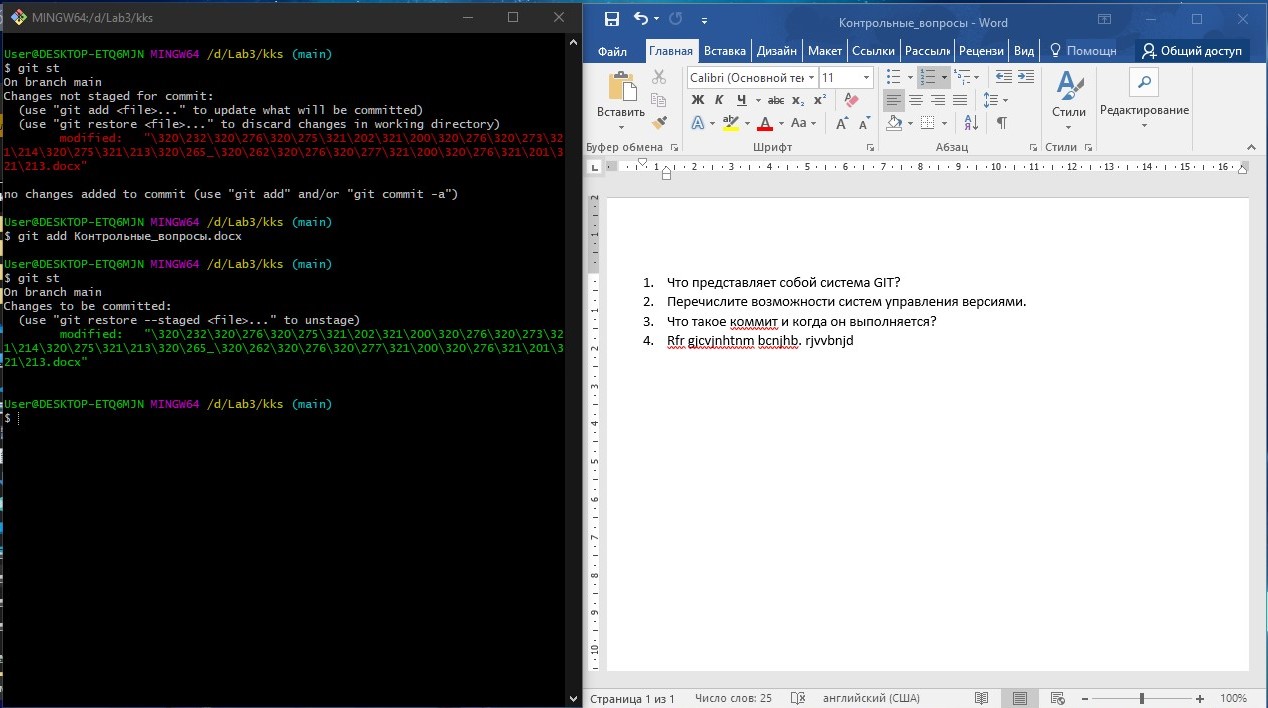


1. Используя команду git tag v.03, создаю тэг для текущей версии проекта. В ходе работы над проектом появляется необходимость откатиться к предыдущей проиндексированной версии, что я делаю с помощью команды

git checkout Контрольные\_вопросы.docx



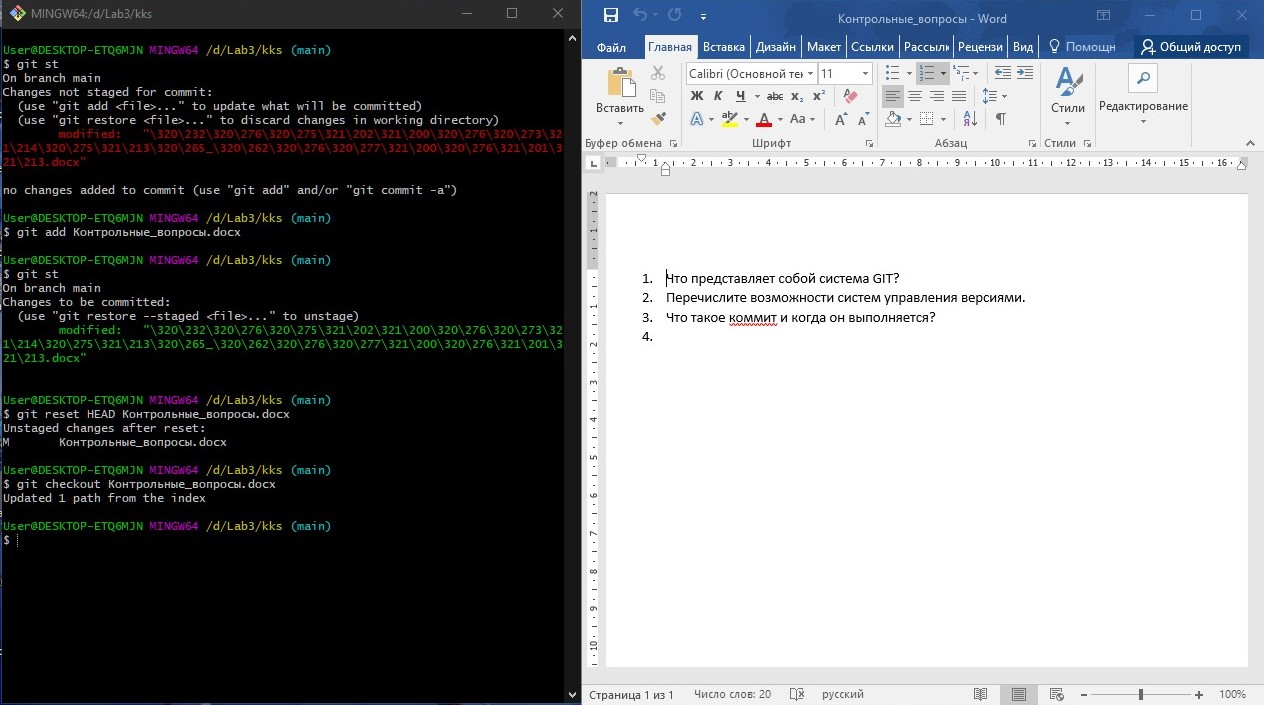
1. В ходе работы над проектом, я случайно проиндексировал файл с ошибками:



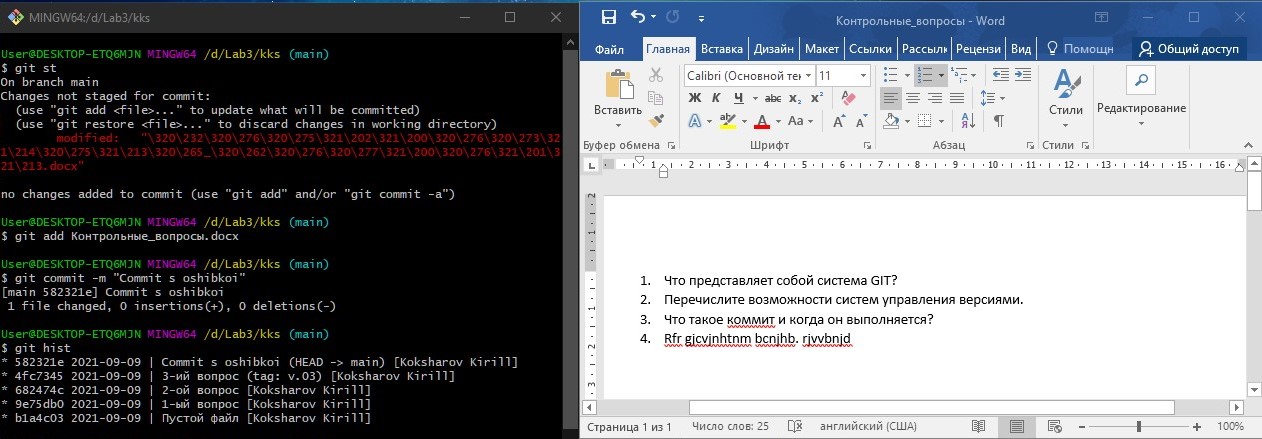
Для решения этой проблемы произвожу откат версии командой:

git reset HEAD Контрольные\_вопросы.docx

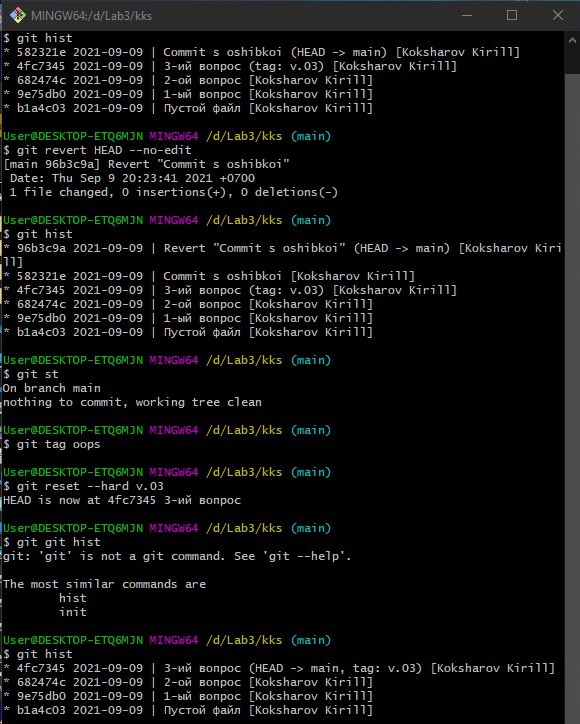
git checkout Конторольные\_вопросы.docx



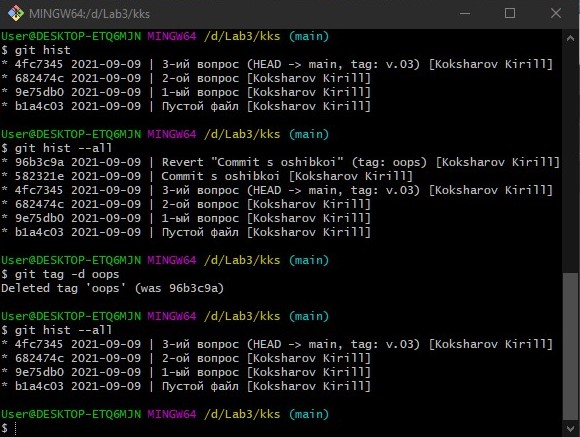
1. Cлучилось так, что файл с ошибками был проиндексирован мной, и к нему добавлен коммит.



Для решения возникшей проблемы произвожу откат командой git revert HEAD –no-edit, тэгирую проблемную версию и командой git reset --hard v.03 выполняю сброс к предыдущей версии.

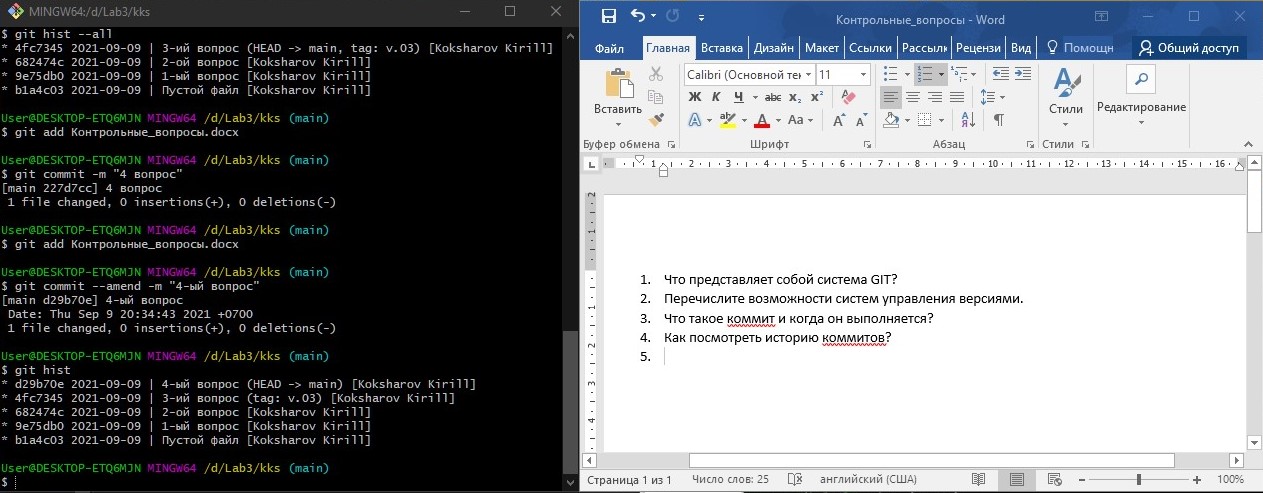


Для полного удаления ошибочной версии произвожу удаление тэга.

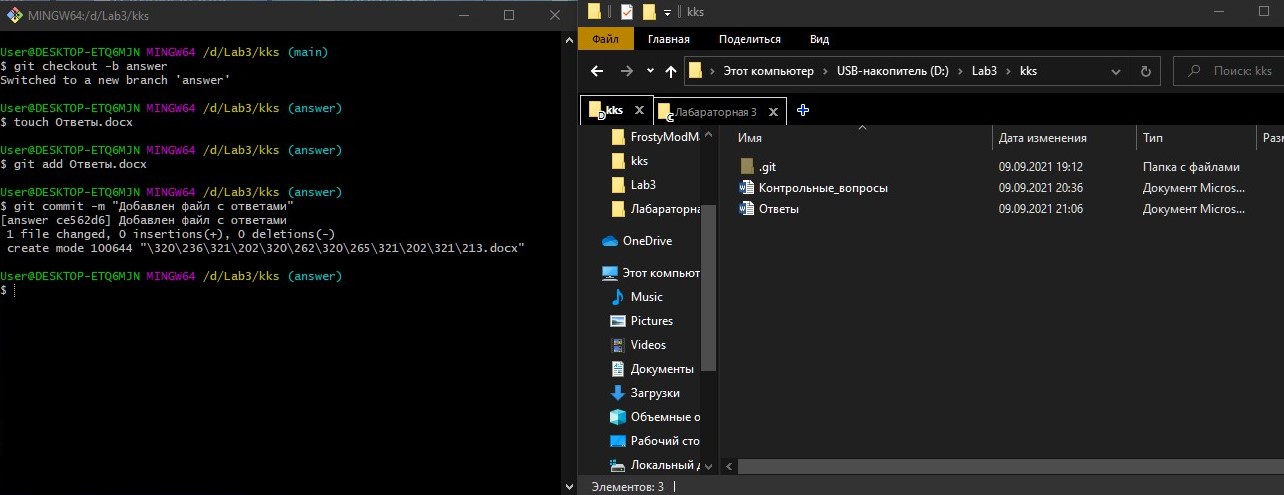


В ходе последующей работы допускаю небольшую ошибку во время коммита, для её исправления использую команду

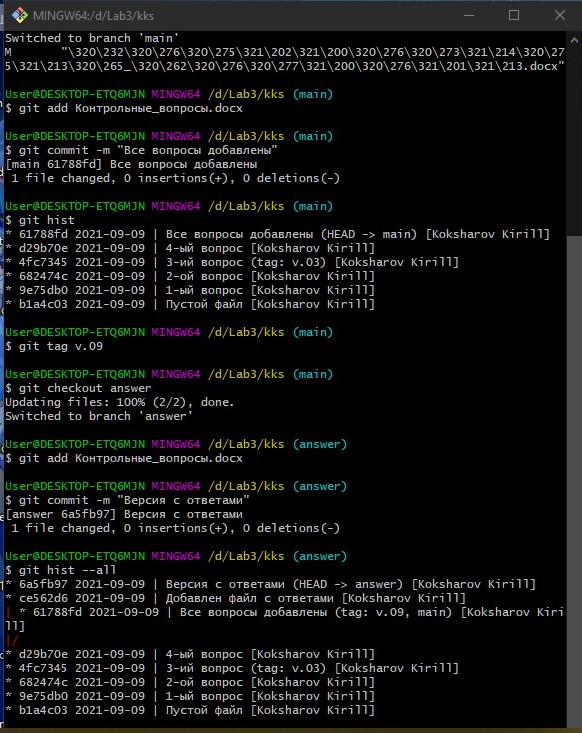
git commit --amend -m “4-ый вопрос”



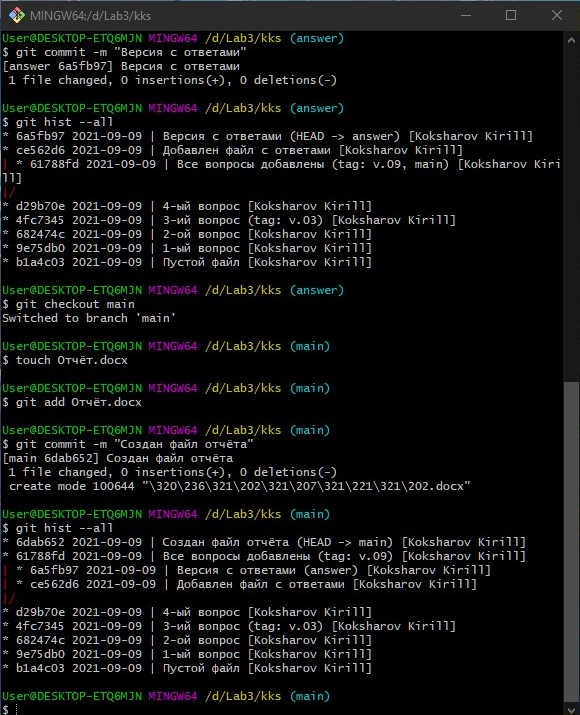
1. Командой git checkout –b answer создаю и перехожу на новую ветку, в которой создаю новый файл Ответы.docx



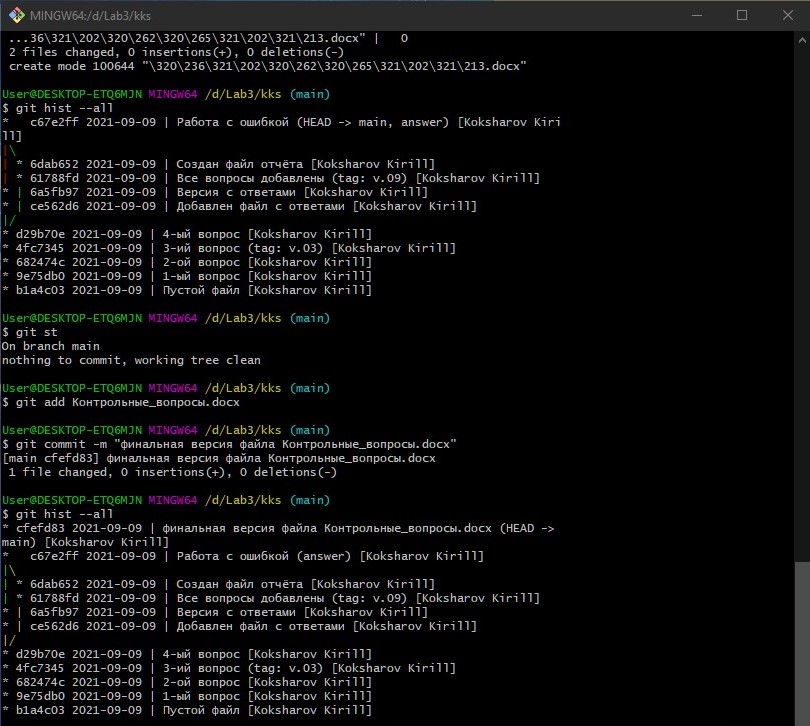
Далее командой git checkout main возвращаюсь на главную ветку, в которой работаю над файлом Контрольные\_вопросы.docx. Затем возвращаюсь на ветку answer (git checkout answer), где также работая над этим файлом.



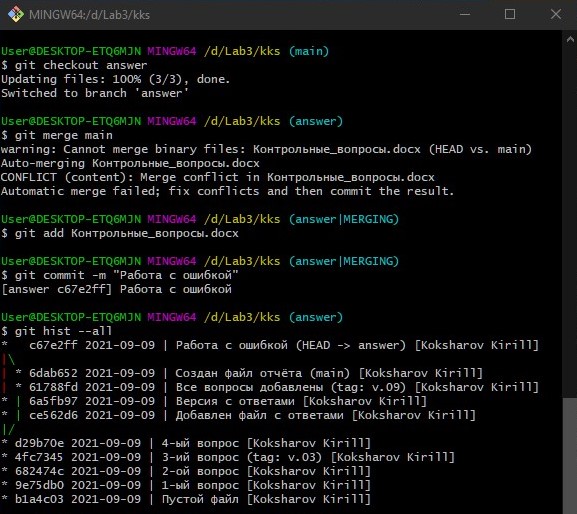
Возвращаюсь на основную ветку, где создаю новый файл Отчёт.docx. Проверяю лог.



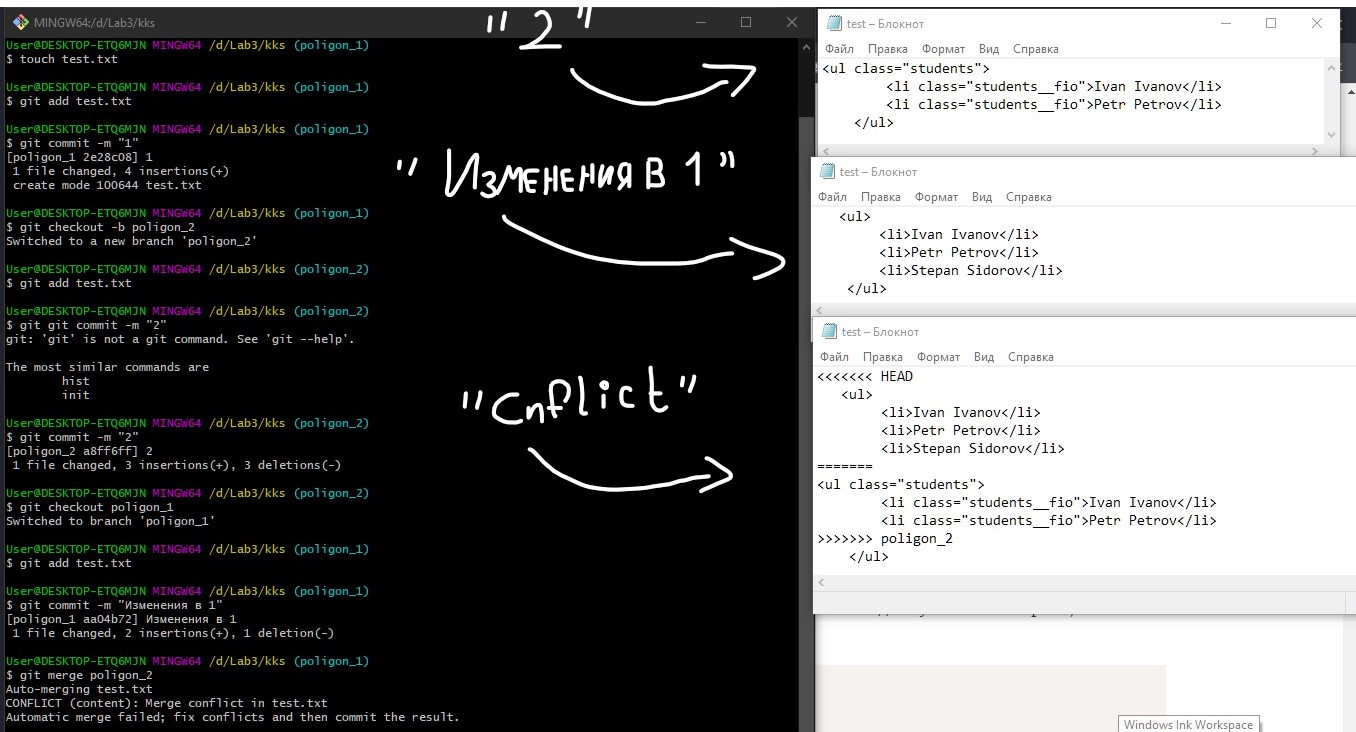
1. После завершения работы с веткой answer решаю провести слияние с веткой main. Для этого использую git merge answer. (В ходе работы возник конфликт, о котором указано в пункте 11)



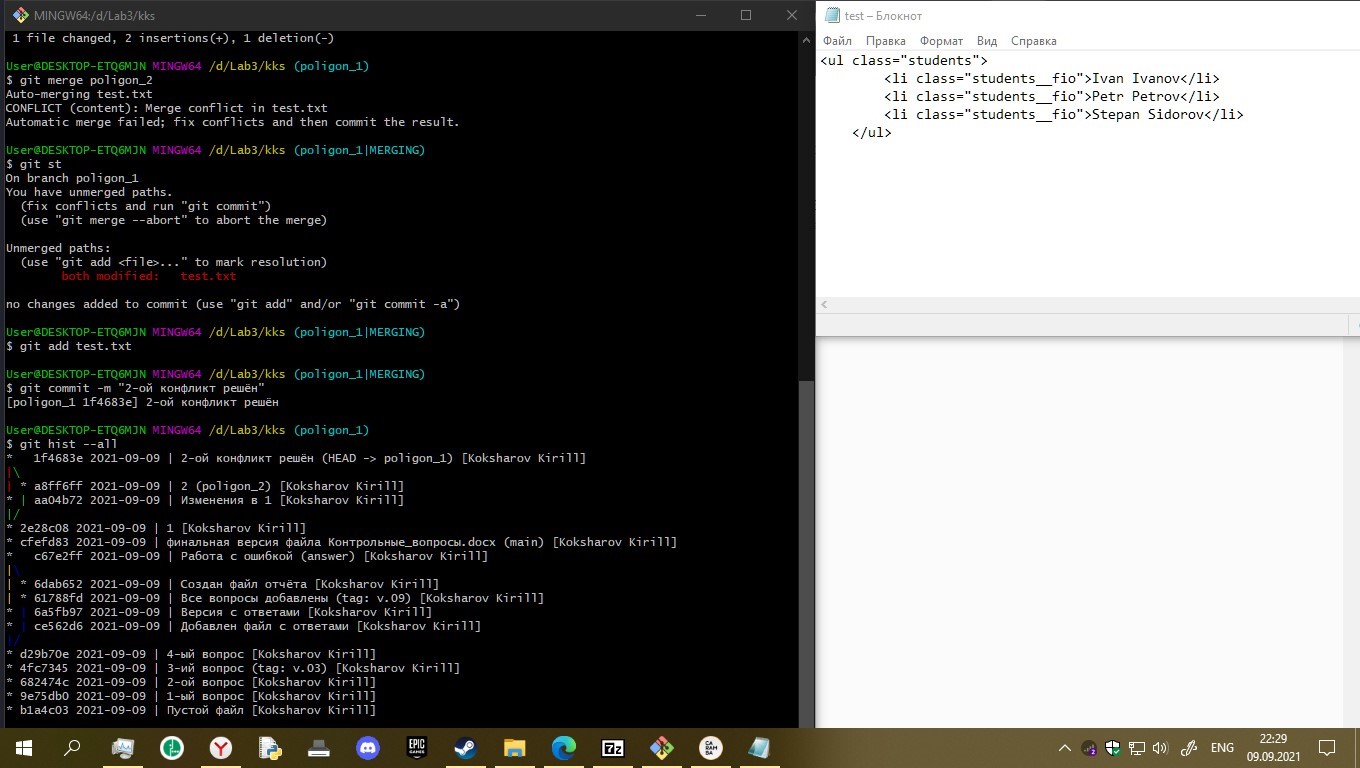
1. В ходе слияния веток случайно возник конфликт, который git не смог решить автоматически. Для решения возникшей ошибки пришлось вносить изменения в файле (привести одну версию файла под другую) после чего инициализировать и закоммитить его.



1. Для лучшего понимания принципов возникновения и решения конфликтов при слиянии решил создать две новые ветви: poligon\_1, где создал файл test.txt, и poligon\_2. Поочерёдно вносил изменения в этот файл в разных ветках после чего решил слить ветку poligon\_2 в poligon\_1.



Ожидаемо, возник конфликт, который git не смог решить сам. Поэтому пришлось вносить изменения в файле самостоятельно, после чего инициализировать и коммитить конфликтный файл.



После завершения работы удалил ветви командами:

git branch –d poligon\_1

git branch –d poligon\_2