Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Сибирский государственный университет науки и технологий

имени академика М.Ф. Решетнева»

**ЛАБОРАТОРНАЯ РАБОТА № 3**

**Знакомство с системой GIT**

10 Вариант

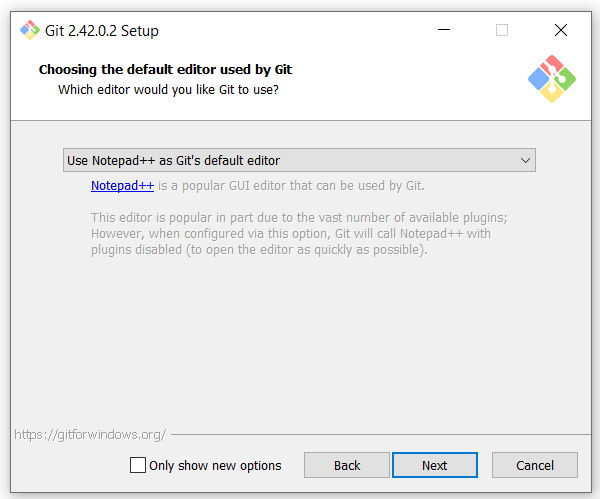
|  |  |
| --- | --- |
|  | Выполнил студент группы БПИ23-01: Симендеева Е.В.  Проверил: Буряченко В.В. |
|  |  |
|  |  |

Красноярск, 2023

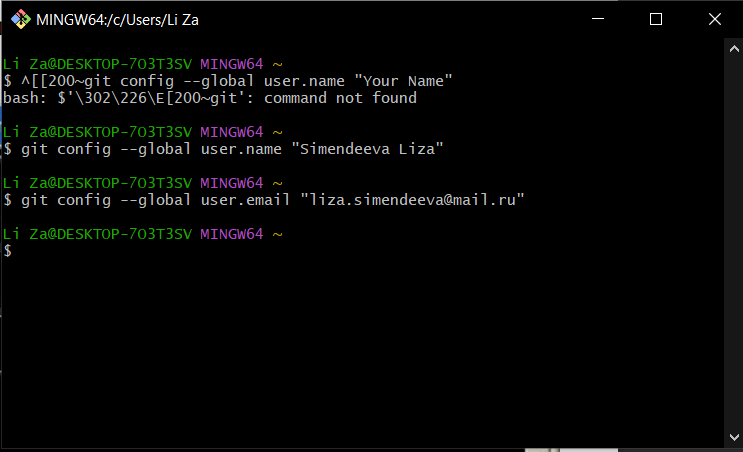
**Задание 10 варианта:** Использовать 3 ветки, создать 2 конфликта и разрешить их.

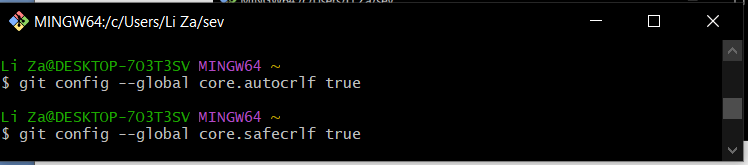
**Ход работы**

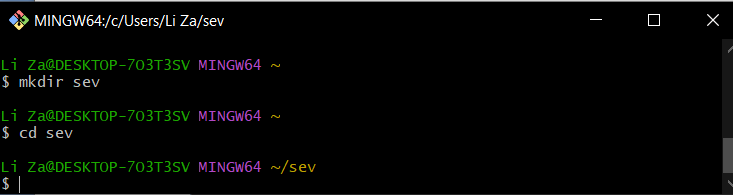
1. Установка GIT

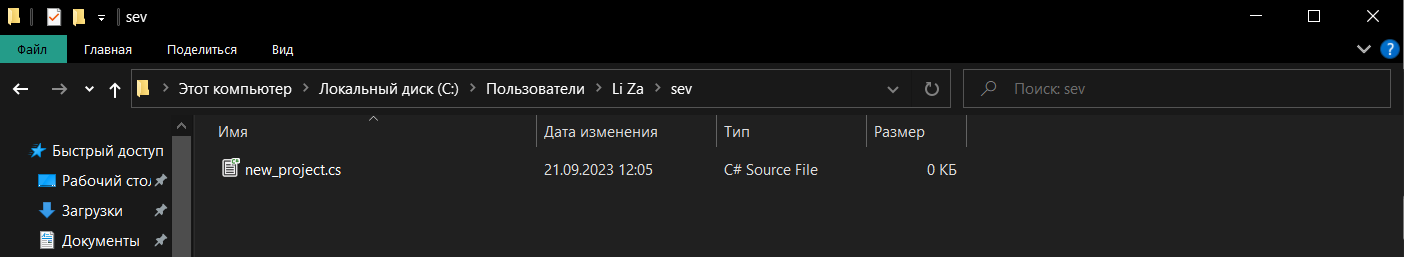
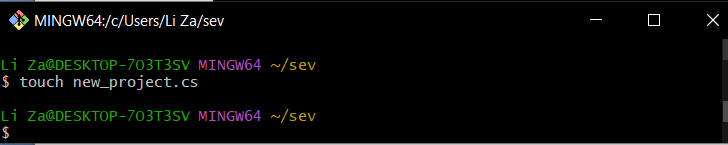


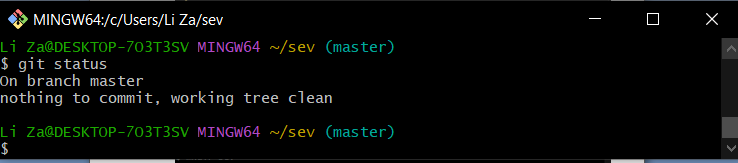
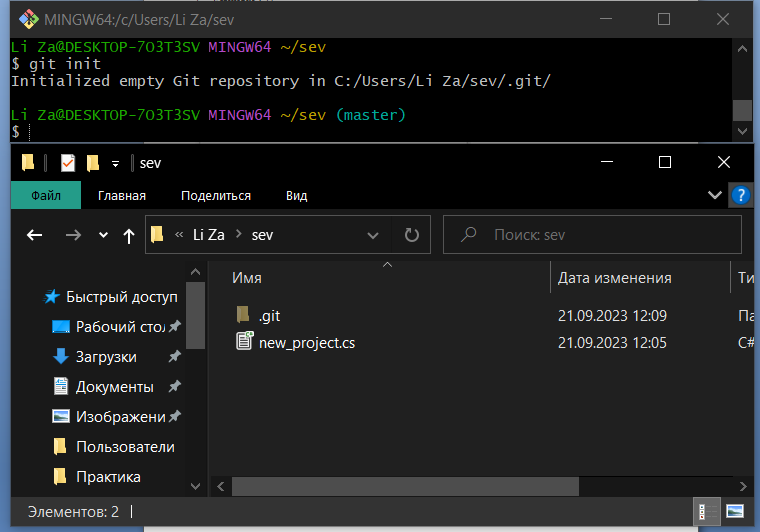
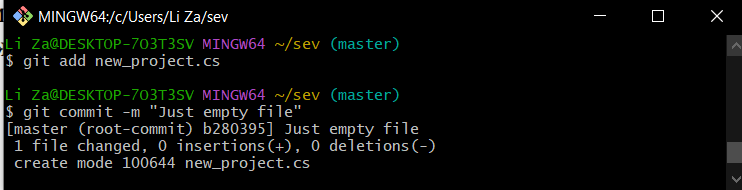
1. Подготовка к работе с Git
   1. Настройка имени и адреса электронной почты

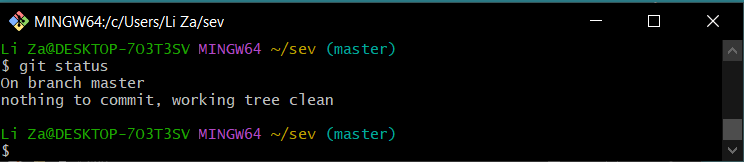


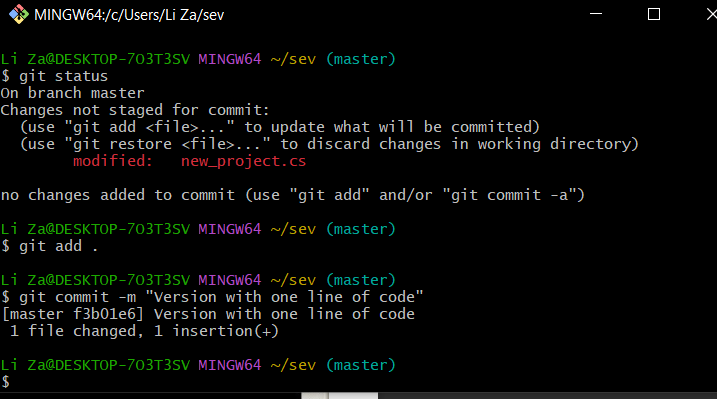
* 1. Настройка параметров окончания строк 

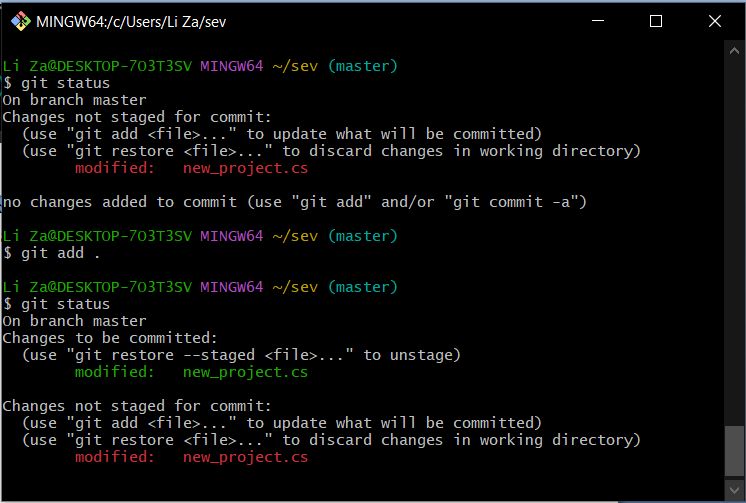
1. Создание проекта и репозитория
   1. Создание каталога 
   2. Создание файла

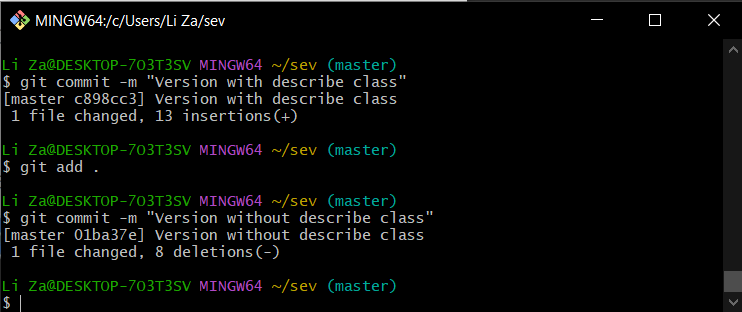


* 1. Проверка состояния репозитория
  2. Создание репозитория
  3. Добавление первого коммита

1. Проверка состояния репозитория
2. Индексация изменений

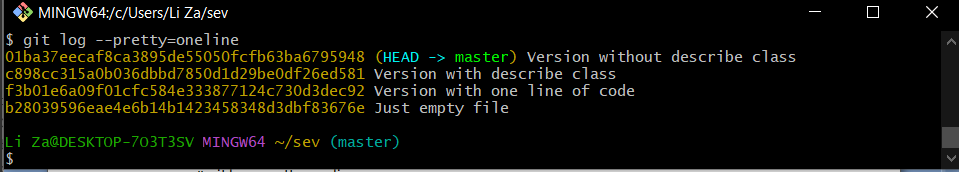
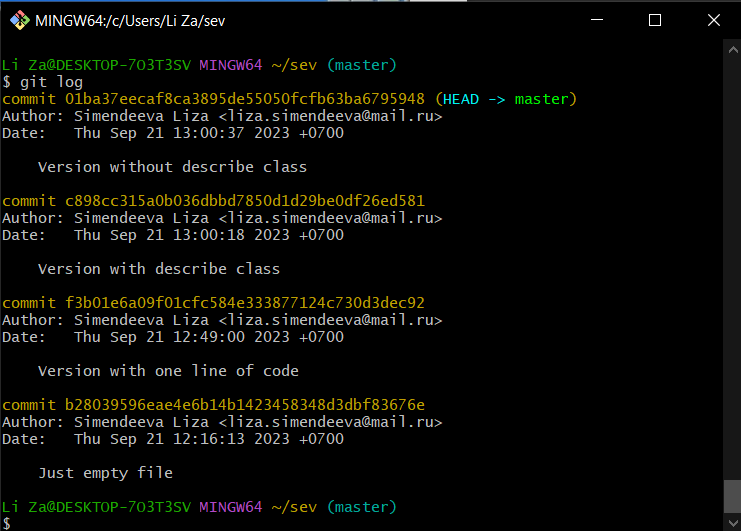
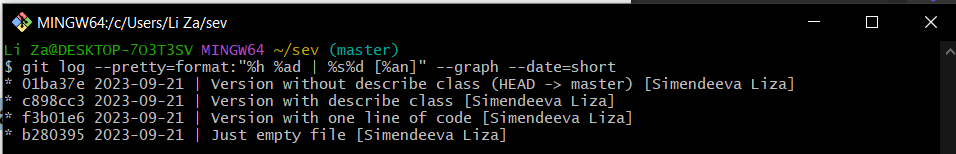
Внесем некоторые изменения в файл проекта и проверим состояние репозитория с помощью git status. Для добавления текущей версии в систему контроля необходимо выполнить индексацию изменений (git add .) и добавить коммит git commit –m <name\_commit>. 

Отредактируем код, добавим изменения в индекс и снова изменим файл, проверим состояние репозитория. Сделаем вывод, что git работает с изменениями, а не с целыми файлами. 

Далее добавим коммит к проиндексированной версии, проиндексируем последнюю версию и добавим коммит к ней.

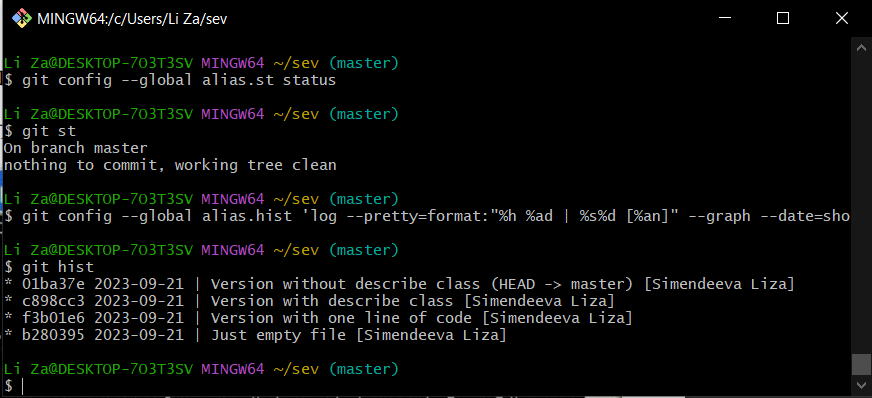
1. История версий

Выведем историю версий с помощью git log. При таком выводе происходит дублирование большого количества информации, поэтому для удобства восприятия истории воспользуемся режимом «одной строки» git log --pretty=oneline

   
Для ещё более удобного вывода истории можно использовать различные способы форматирования. 

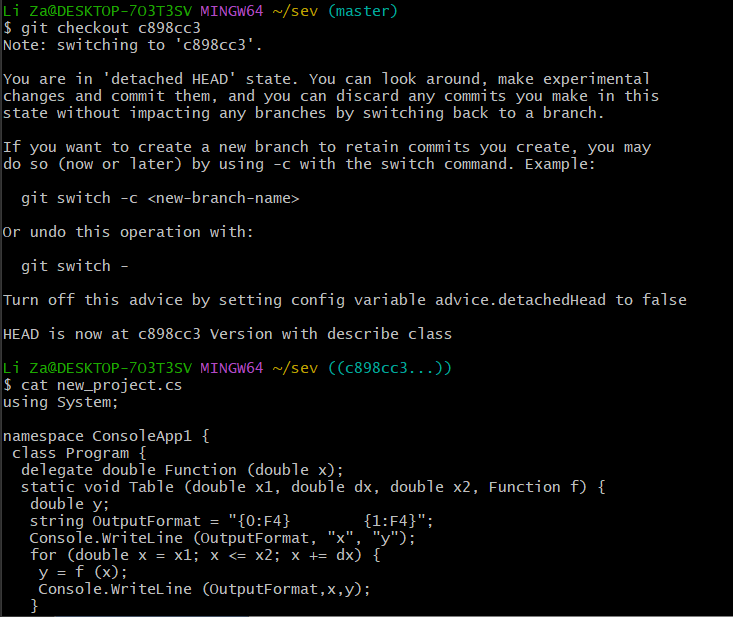
1. Алиасы

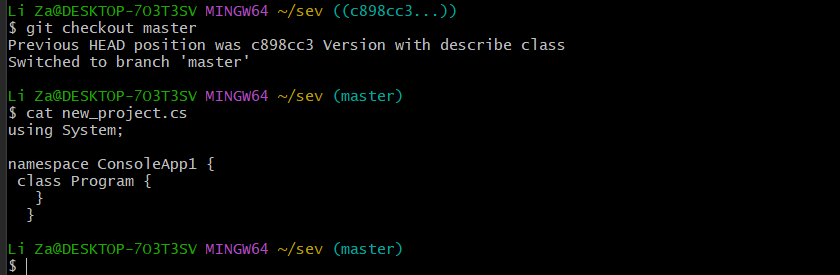
Сокращаем команды git status и git log --pretty=format:"%h %ad | %s%d [%an]" --graph --date=short с помощью использования алиасов.



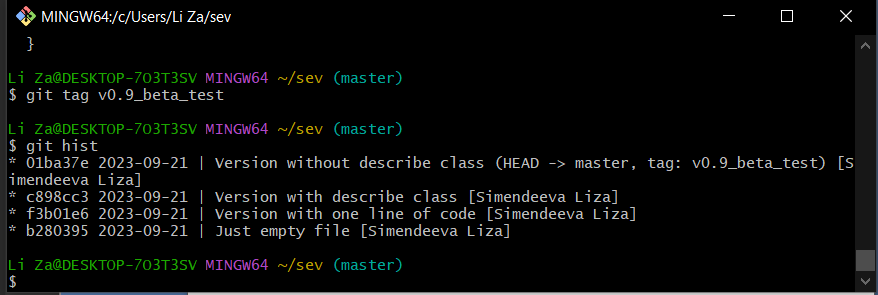
1. Возврат к прошлым версиям

Вернуться к прошлой версии кода программы возможно с помощью git checkout <hash>, последней – git checkout master



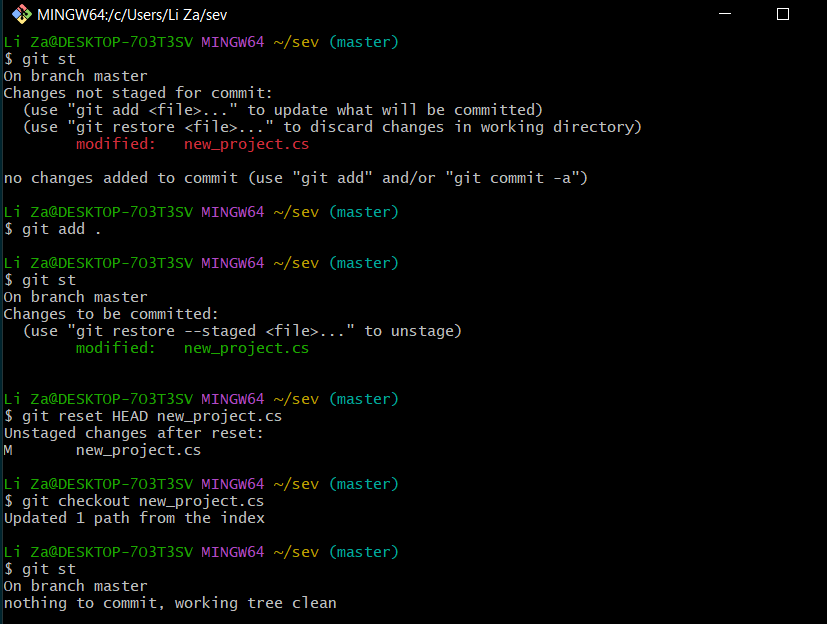


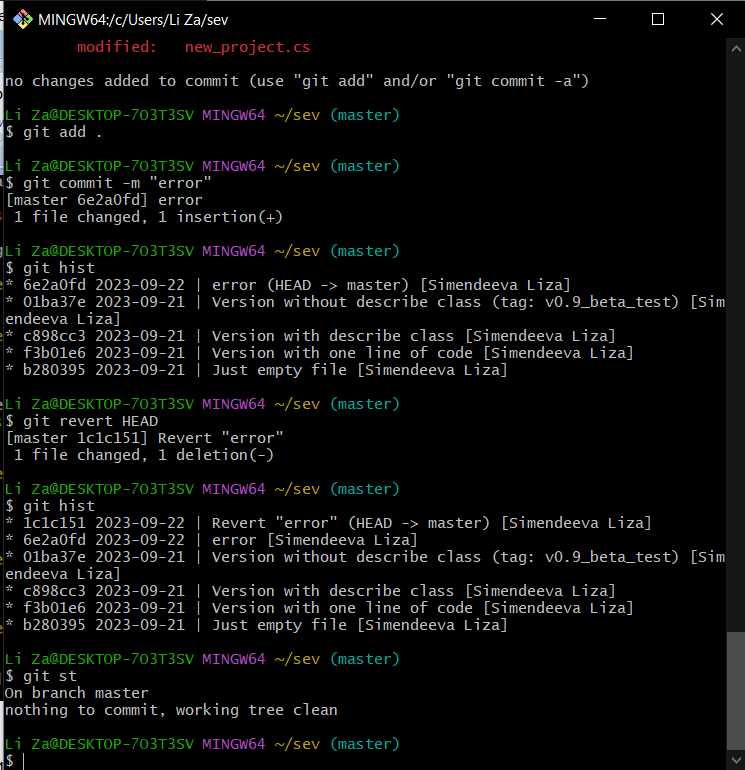
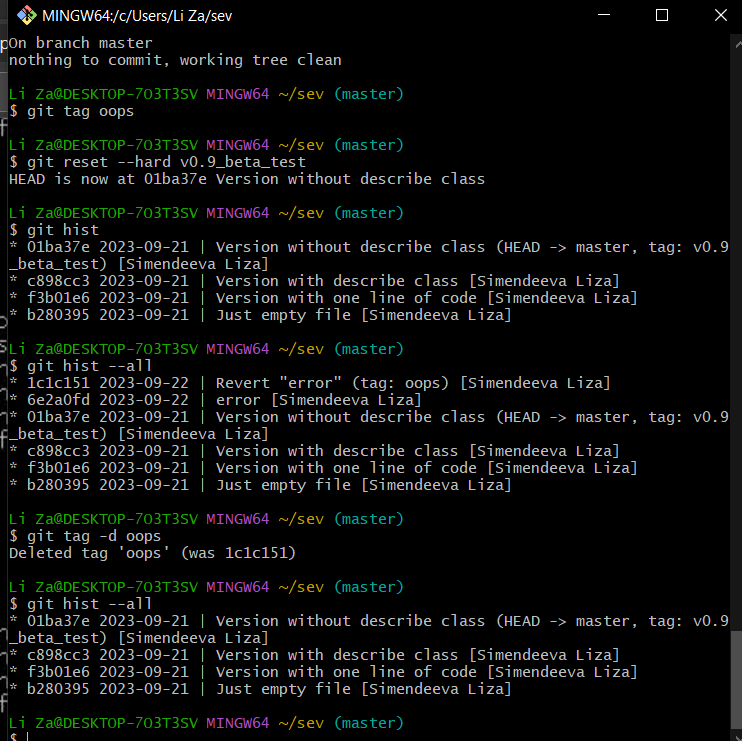
1. Тегирование версий

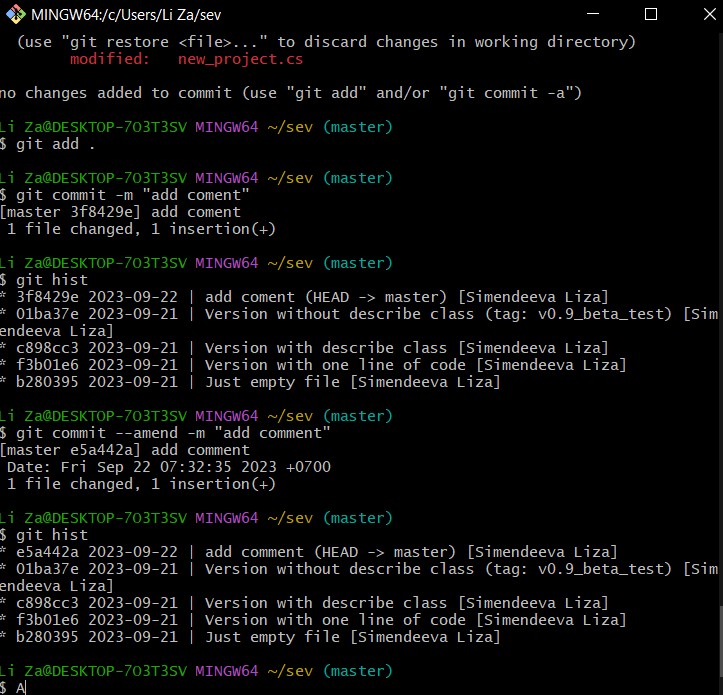
Когда разработка проекта ведется в несколько этапов, целесообразно использовать тегирование с помощью команды git tag <some\_tag>.

1. Отмена локальных изменений

С помощью git checkout <file\_name> можно вернуться на последнюю проиндексированную версию.

  
  
Добавим файл с изменениями в индекс и с помощью git reset HEAD <file\_name> откатим версию обратно. 

Добавим файл в индекс и коммит, откатим изменения, выполнив команду git revert HEAD.   
  
  
  
Отмененный коммит остался в истории, для его отмены для необходимо протегировать текущую версию файла. После этого выполнить сброс к предыдущей версии.  


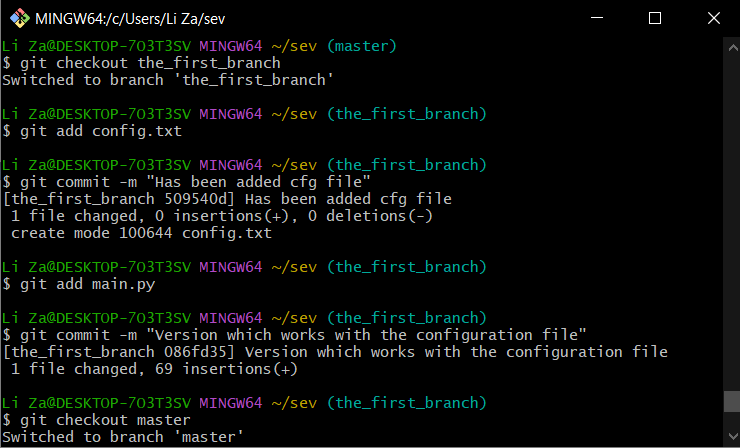
Изменение названия коммита с помощью git commit –amend –m “name\_commit”

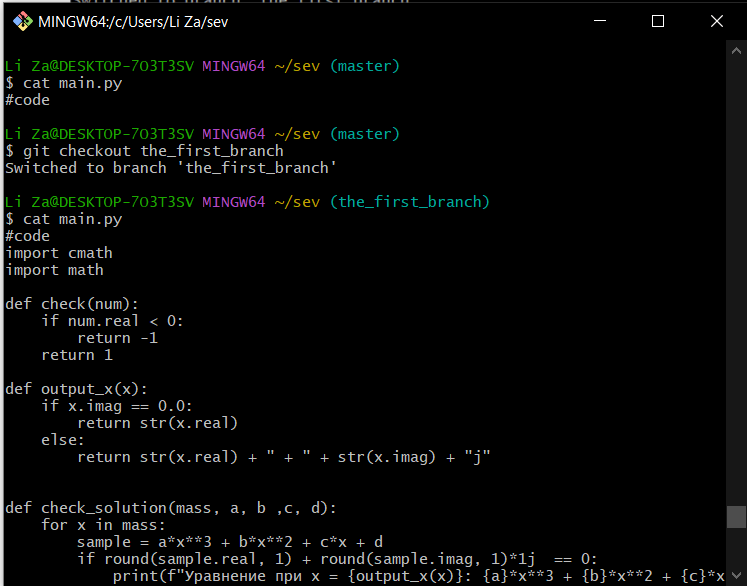
1. Создание ветки

Создадим три ветки, каждая из которых будет предназначена для редактирования отдельных файлов: main.py, config.txt, README.

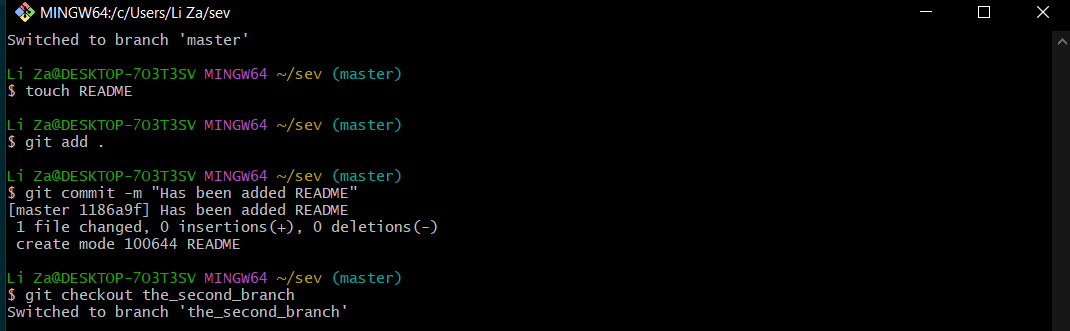


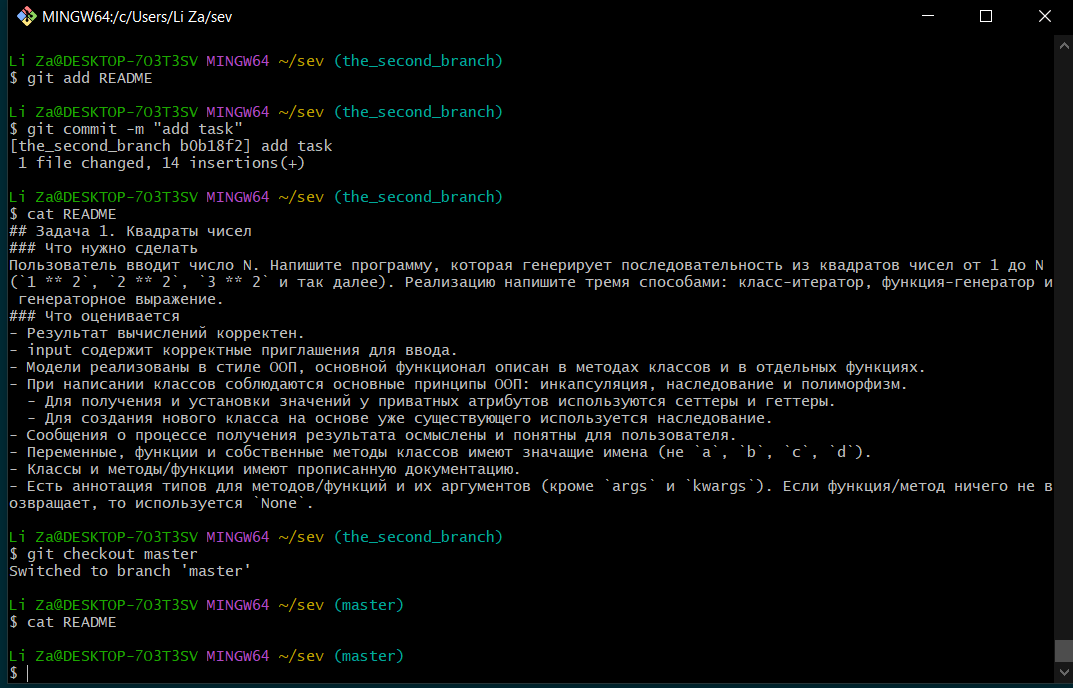
Перейдем на первую ветку, изменим main.py и сделаем коммит.



В основной ветке файл main.py не изменился. 

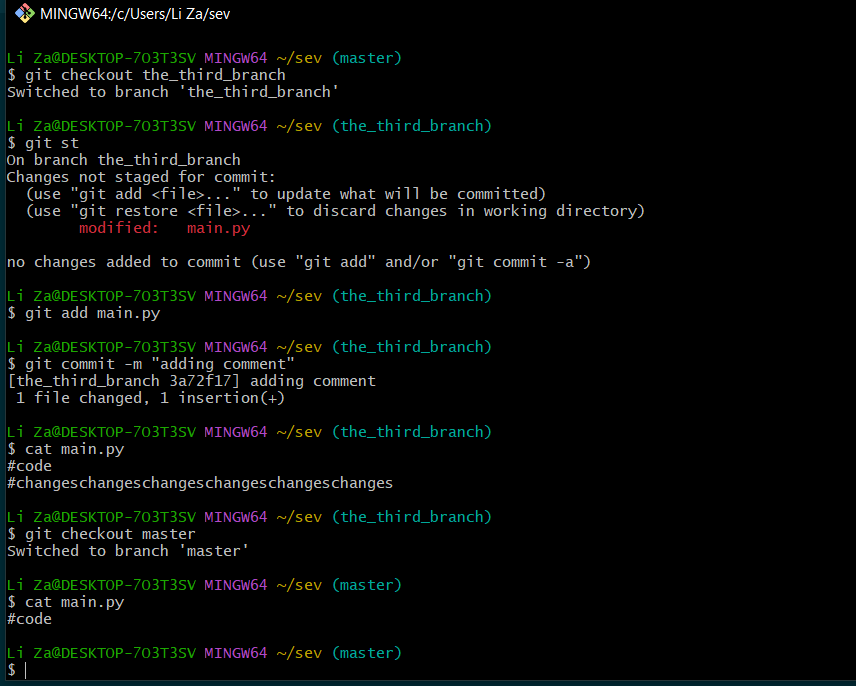
Выполним аналогичные действия с другими ветками. Заметим, что config.txt нет во второй ветки, так как файл создан в первой ветке.



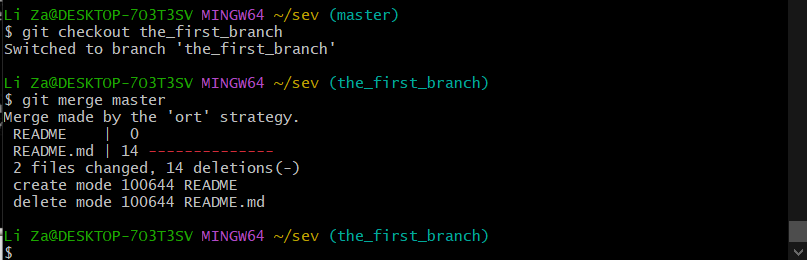


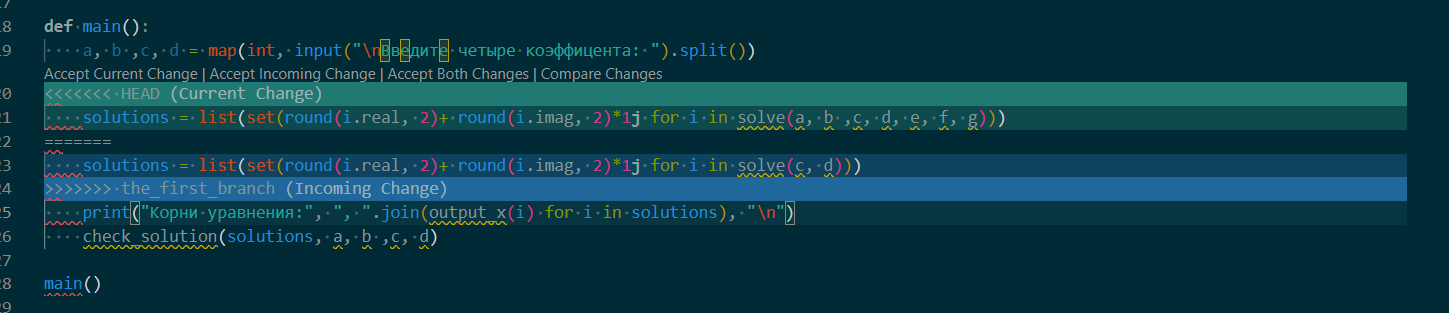
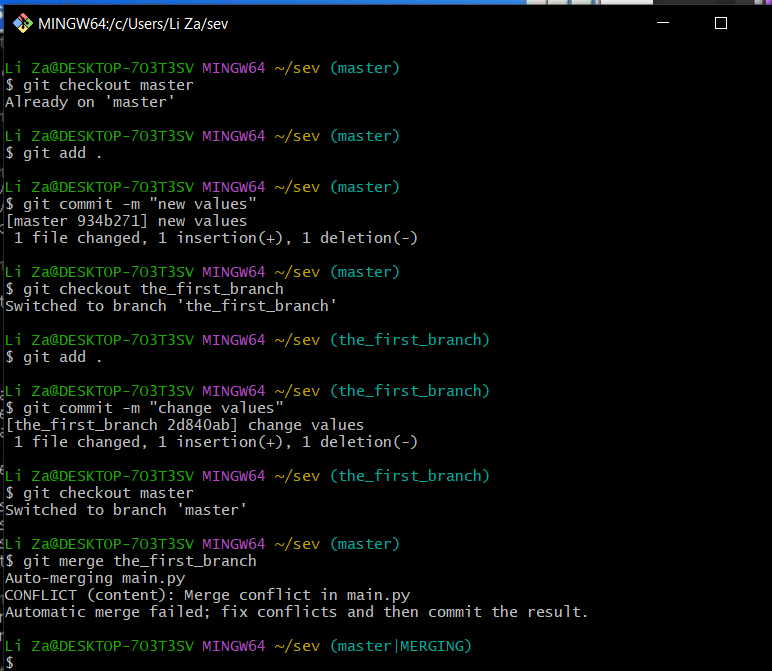
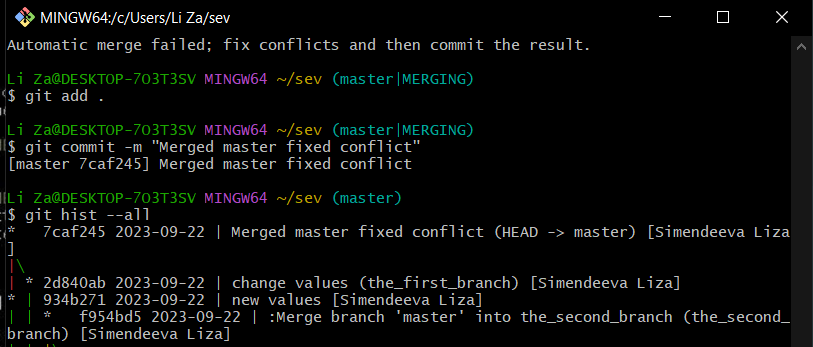
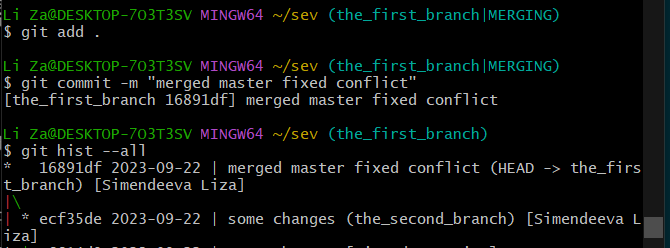
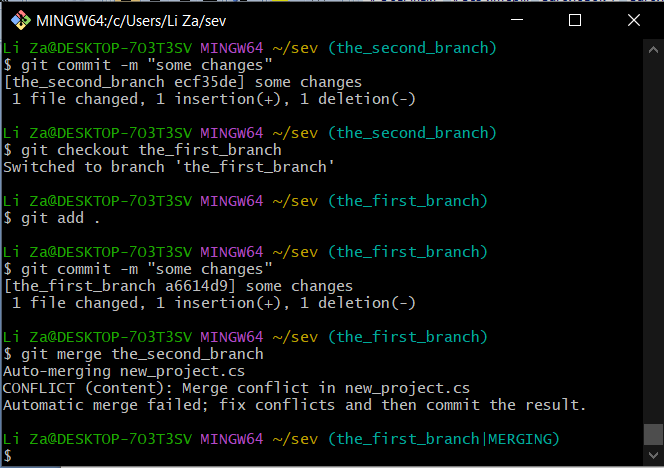
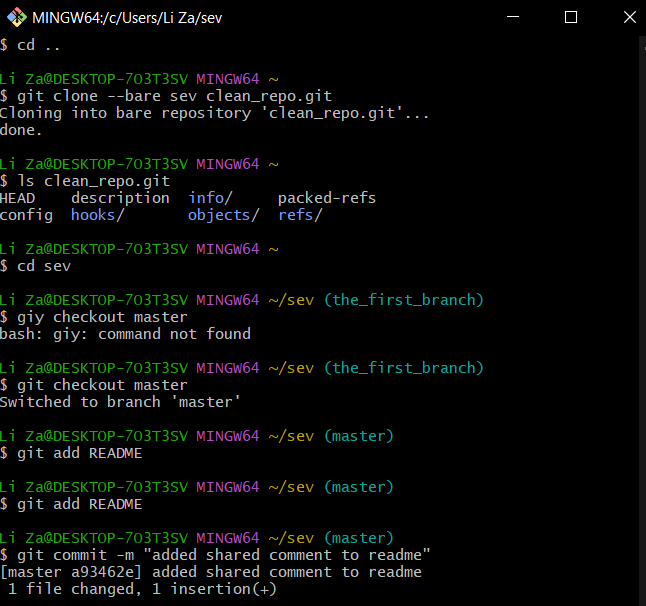
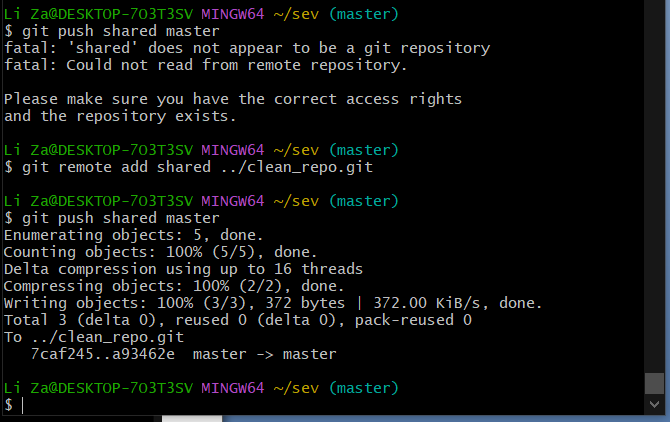
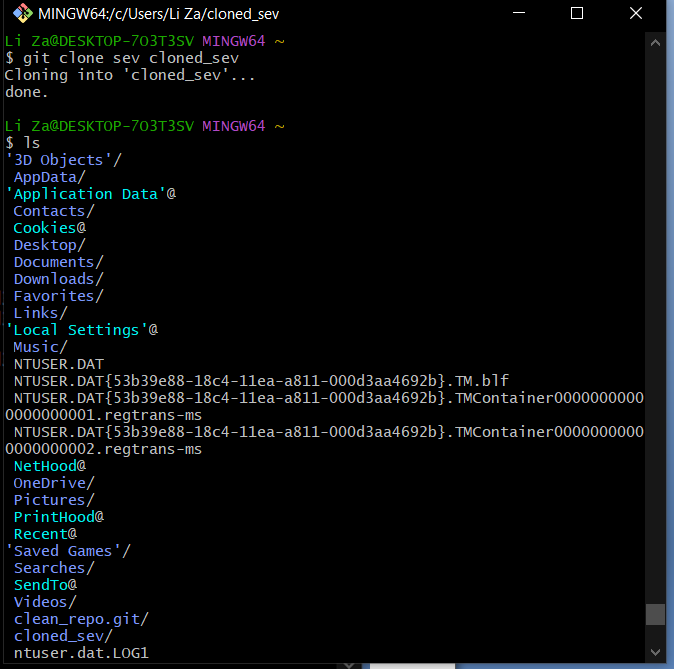
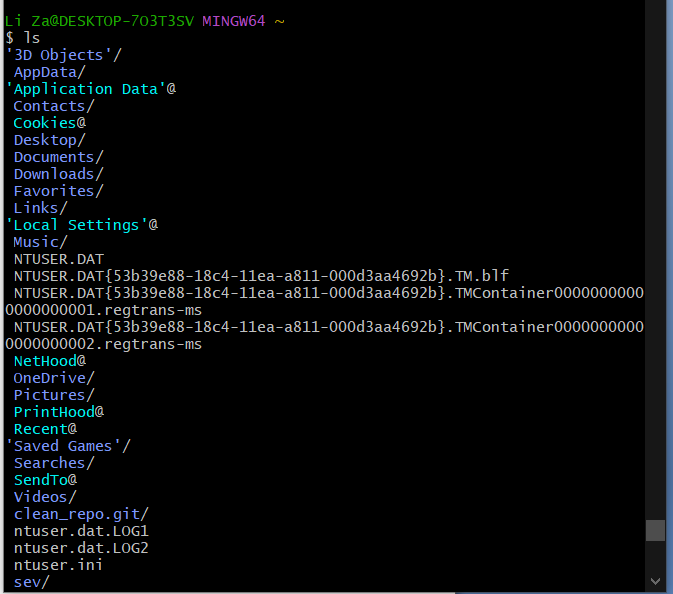


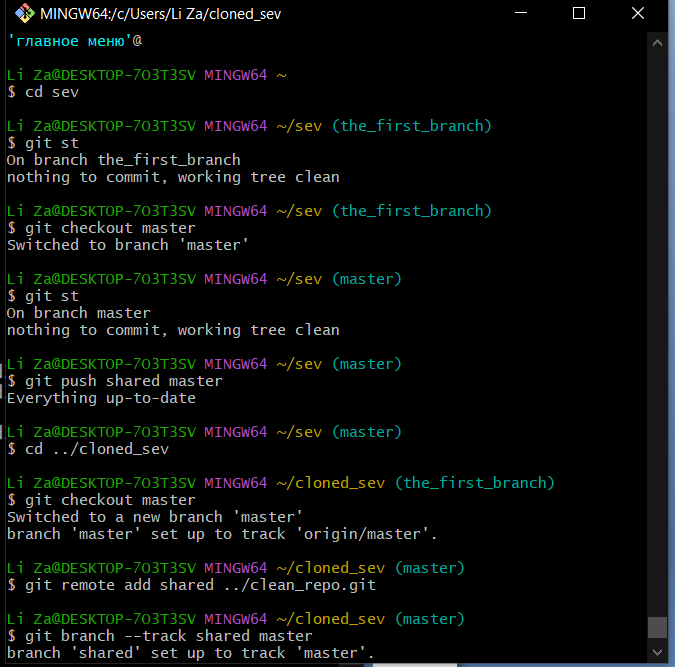
main.py остался неизменным в ветке master.

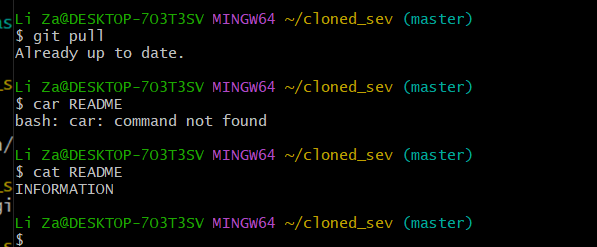


1. Слияние веток



1. Конфликт
2. Разрешение конфликтов1  
   Ещё раз создание и разрешение конфликта. 
3. Создание чистого репозитория  
   Создадим копию удаленного репозитория командой git clone –bare <repo\_name> <new\_repo\_name>, добавим удалённый репозиторий и присвоим ему имя, выполнив команду git remote add <shortname> <url>. Изменим файл и отправим изменения с помощью git push <remote-name> <branch-name>  
     
     
     
     
     
     
     
     
     
     
     
     
     
     
   Просмотрим список файлов командой ls. Скопируем репозиторий git clone sev cloned\_sev.   
     
     
     
     
     
     
     
     
     
     
     
   Выполним новые команды:

git branch –track shared master: здесь будет создана новая локальная ветка "shared", которая будет отслеживать удаленную ветку "master".  
git pull: объединяет изменения из удаленного репозитория в текущую ветку локального репозитория  




**Вывод**

В ходе прохождения лабораторной работы были выполнены основные виды работ по управлению версиями ПО с использованием системы контроля версий GIT. Данная система контроля версий позволяет совместно работать над проектом: отслеживать, кем и когда внесены изменения в код, предоставляет возможность откатиться к предыдущим версиям кода. Также GIT предоставляет инструменты для ветвления и слияния кода, позволяет работать в отдельных ветках.

**Контрольные вопросы**

1. Что представляет собой система GIT?  
   Система контроля версий (Version Control System, VCS) – это ПО, позволяющее отслеживать изменения в документах, при необходимости производить их откат, определять, кто и когда внес исправления и т.п.
2. Перечислите возможности систем управления версиями.  
   VCS отслеживает изменения в файлах, предоставляет возможности для создания новых и слияние существующих ветвей проекта, производит контроль доступа пользователей к проекту, позволяет откатывать исправления и определять кто, когда и какие изменения вносил в проект.
3. Что такое коммит и когда он выполняется?  
   Commit – это операция отправки изменений в репозиторий. Он выполняется после того, как решено, что все нужные изменения на данный момент внесены, и об этом можно сообщить системе контроля версий. Сначала изменения следует проиндексировать с помощью add, после использовать commit.
4. Как посмотреть историю коммитов?  
   Историю коммитов можно просмотреть с помощью команд git log, можно использовать различные способы форматирования, например:   
   git log --pretty=oneline
5. Как создать пустой GIT репозиторий?  
   Создать каталог mkdir <name-dir>, создать файл touch <name-file>, создать репозиторий git init
6. Какую архитектуру имеет система контроля версий GIT?

Система контроля версий GIT использует распределённую архитектуру: каждый рабочий каталог содержит полную историю всех изменений проекта, каждый разработчик имеет копию репозитория.

1. Как создать новую ветку в GIT и соединить ветки?  
   За создание веток отвечает команда git branch <branch\_name>, слияние git merge <branch\_name>
2. Поясните понятие «staging area».

Staging area (index) – промежуточная область в Git, которая служит для подготовки и организации изменений, которые будут входить в следующий коммит.

1. Как сравнить версии файла в GIT?

git diff <file\_name>: сравнить изменения между рабочей директорией и индексом  
git diff <commit1> <commit2> <file\_name>: сравнить версии файла между двумя коммитами  
git difftool -t meld <file\_name>: откроет Meld для сравнения версий файла