|  |
| --- |
|  |
| МИНОБРНАУКИ РОССИИ |
| *Федеральное государственное бюджетное образовательное учреждение высшего образования* ***«МИРЭА – Российский технологический университет»***  **РТУ МИРЭА** |

Институт Информационных технологий (ИТ)

Кафедра Математического обеспечения и стандартизации информационных технологий (МОСИТ)

|  |
| --- |
| **ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ № 1** |
| **по дисциплине** |
| **«Технология разработки программных приложений»**  **Тема: «Системы контроля версий»** |

|  |  |
| --- | --- |
| Выполнил студент группы \_\_\_\_\_ИКБО-03-21\_ | Хречко С.В, |
| Принял ассистент кафедры МОСИТ | Петрова А. А. |

|  |  |  |
| --- | --- | --- |
| Практическая работа выполнена | «\_\_»\_\_\_\_\_\_\_2023\_г. | *(подпись студента)* |
|  |  |  |
| «Зачтено» | «\_\_»\_\_\_\_\_\_\_2023\_ г. | *(подпись руководителя)* |

Москва 2023г.

Оглавление

[Задачи 3](#_Toc128422926)

[Часть первая 6](#_Toc128422927)

[Пункт 1 6](#_Toc128422928)

[Пункт 2 6](#_Toc128422929)

[Пункт 3 7](#_Toc128422930)

[Пункт 4 7](#_Toc128422931)

[Пункт 5 8](#_Toc128422932)

[Пункт 6 9](#_Toc128422933)

[Пункт 7 10](#_Toc128422934)

[Пункт 8 11](#_Toc128422935)

[Пункт 9 12](#_Toc128422936)

[Пункт 10 13](#_Toc128422937)

[Пункт 11 14](#_Toc128422938)

[Часть вторая 14](#_Toc128422939)

[Пункт 1 14](#_Toc128422940)

[Пункт 2 15](#_Toc128422941)

[Пункт 3 16](#_Toc128422942)

[Пункт 4 16](#_Toc128422943)

[Пункт 5 17](#_Toc128422944)

[Пункт 6 17](#_Toc128422945)

[Пункт 7 18](#_Toc128422946)

[Подпункт 1 18](#_Toc128422947)

[Подпункт 2 18](#_Toc128422948)

[Подпункт 3 18](#_Toc128422949)

[Подпункт 4 19](#_Toc128422950)

[Подпункт 5 20](#_Toc128422951)

[Подпункт 6 20](#_Toc128422952)

[Подпункт 7 21](#_Toc128422953)

[Часть третья 21](#_Toc128422954)

[Пункт 1 21](#_Toc128422955)

[Пункт 2 22](#_Toc128422956)

[Пункт 3 23](#_Toc128422957)

[Пункт 4 23](#_Toc128422958)

[Пункт 5 23](#_Toc128422959)

[Пункт 6 24](#_Toc128422960)

[Пункт 7 25](#_Toc128422961)

[Пункт 8 26](#_Toc128422962)

[Пункт 9 26](#_Toc128422963)

[Пункт 10 27](#_Toc128422964)

[Пункт 11 27](#_Toc128422965)

[Пункт 12 27](#_Toc128422966)

[Ответы на контрольные вопросы 28](#_Toc128422967)

[Вопрос 2 28](#_Toc128422968)

[Вопрос 3 28](#_Toc128422969)

[Вопрос 4 29](#_Toc128422970)

[Вопрос 5 29](#_Toc128422971)

[Вопрос 10 29](#_Toc128422972)

[Вопрос 18 30](#_Toc128422973)

[Вопрос 27 30](#_Toc128422974)

[Вопрос 28 30](#_Toc128422975)

**Цель работы**: изучить структуру и функционал рассматриваемой информационной системы.

# Задачи

**Часть 1. Основные команды Git**

1. Установите и настройте клиент git на своей рабочей станции.

2. Создайте локальный репозиторий и добавьте в него несколько файлов.

3. Внесите изменения в один из файлов.

4. Проиндексируйте изменения и проверьте состояние.

5. Сделайте коммит того, что было проиндексировано в репозиторий. Добавьте к коммиту комментарий.

6. Измените еще один файл. Добавьте это изменение в индекс git. Измените файл еще раз. Проверьте состояние и произведите коммит проиндексированного изменения. Теперь добавьте второе изменение в индекс, а затем проверьте состояние с помощью команды git status. Сделайте коммит второго изменения.

7. Просмотрите историю коммитов с помощью команды git log. Ознакомьтесь с параметрами команды и используйте некоторые из них для различного формата отображения истории коммитов.

8. Верните рабочий каталог к одному из предыдущих состояний.

9. Изучите, как создавать теги для коммитов для использования в будущем.

10. Отмените некоторые изменения в рабочем каталоге (до и после индексирования).

11. Отмените один из коммитов в локальном репозитории.

**Часть 2. Системы управления репозиториями**

1. Создайте аккаунт на GitHub (у кого нет),

2. Создайте репозиторий на GitHub и на локальной машине, согласно выбранной теме проекта,

3. Создайте несколько файлов на локальной машине при помощи консоли,

4. Создайте SSH-ключ для авторизации,

5. Свяжите репозиторий локальной машины с репозиторием на GitHub при помощи консоли,

6. Создайте новую ветку в репозитории с помощью команды, произведите в ней какие-нибудь изменения, а после слейте с веткой master,

7. Выполните цепочку действий в репозитории, согласно вариантам.

Вариант № 5

Задание:

1) Клонируйте непустой удаленный репозиторий на локальную машину

2) Создайте новую ветку и выведите список всех веток

3) Произведите 3 коммита в новой ветке в разные файлы

4) Выгрузите изменения в удаленный репозиторий

5) Произведите revert предпоследнего коммита в новой ветке

6) Выведите в консоли различия между веткой master и новой веткой

7) Слейте новую ветку с master при помощи merge

**Часть 3. Работа с ветвлением и оформление кода**

1. Сделайте форк репозитория в соответствии с вашим вариантом

2. Склонируйте его на локальную машину

3. Создайте две ветки branch1 и branch2 от последнего коммита в master'е

4. Проведите по 3 коммита в каждую из веток, которые меняют один и тот же кусочек файла

5. Выполните слияние ветки branch1 в ветку branch2, разрешив конфликты при этом

6. Выгрузите все изменения во всех ветках в удаленный репозиторий

7. Проведите еще 3 коммита в ветку branch1

8. Склонируйте репозиторий еще раз в другую директорию

9. В новом клоне репозитории сделайте 3 коммита в ветку branch1

10. Выгрузите все изменения из нового репозитория в удаленный репозиторий

11. Вернитесь в старый клон с репозиторием, выгрузите изменения с опцией -force

12. Получите все изменения в новом репозиторииВариант № 7

Репозиторий: <https://github.com/kettanaito/naming-cheatsheet>

# Часть первая

## Пункт 1

В первую очередь устанавливаем и настраиваем Git. На рисунке 1 представлено окно окончания установки Git.

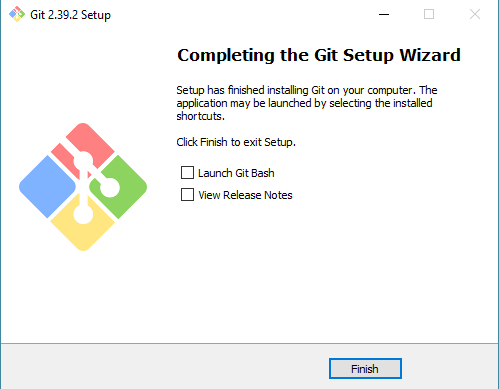


Рисунок 1 – Установка Git

Далее проведем первоначальную настройку Git. На рисунке 2 представлена первоначальная настройка, в частности настройка имени и почты.

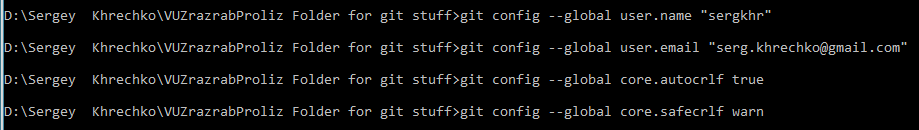


Рисунок 2 – Первоначальная настройка

## Пункт 2

На рисунке 3 представлено создание репозитория в папке hello и добавление в него нескольких файлов.

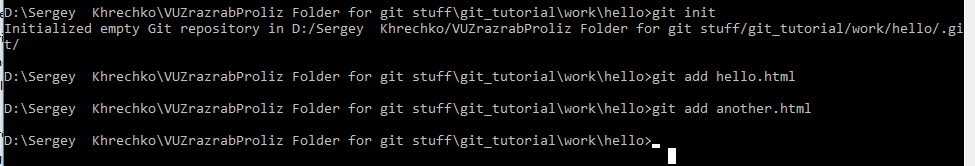


Рисунок 3 – Создание и наполнение репозитория

## Пункт 3

На рисунке 4 представлено внесение изменений в файл hello.html.

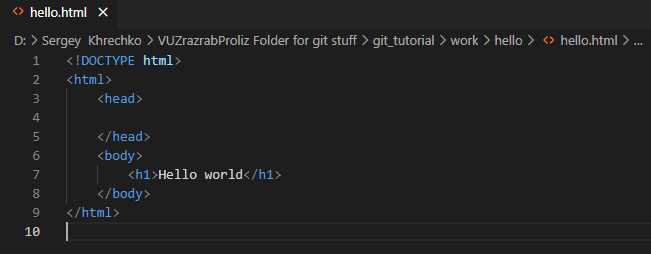


Рисунок 4 – Внесение изменений в файл

На рисунке 5 показано, что изменения еще не проиндексированы.

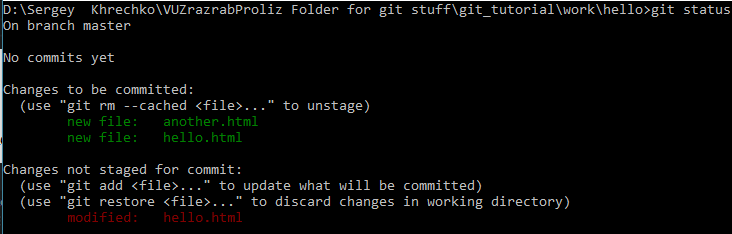


Рисунок 5 – Не проиндексированные изменения

## Пункт 4

На рисунке 6 представлено индексирование изменений в файле hello.html и проверка статуса репозитория.

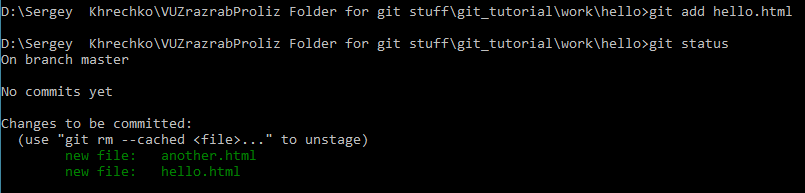


Рисунок 6 – Индексирование и проверка

## Пункт 5

На рисунке 7 представлено совершение коммита, а также добавление комментария.

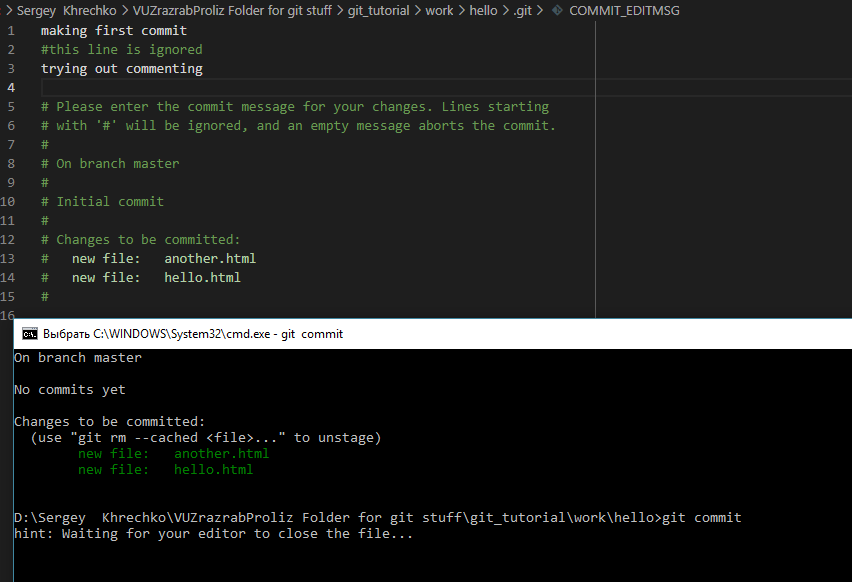


Рисунок 7 – Коммит и комментарий

На рисунке 8 представлены результаты первого коммита.

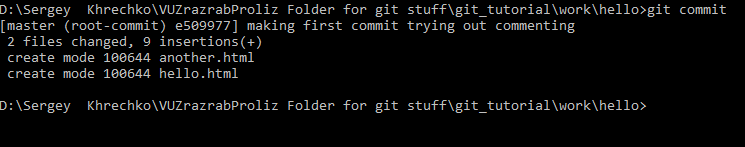


Рисунок 8 – Результаты

## Пункт 6

На рисунке 9 представлено внесение изменений в другой файл и их индексирование.

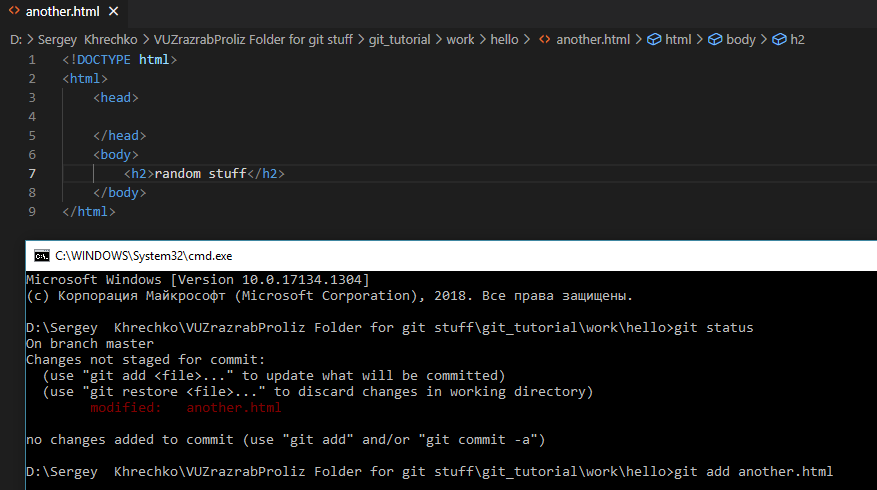


Рисунок 9 – Изменение и индексирование

На рисунке 10 представлено индексирование изменений, а также проверка репозитория после внесения новых изменений.

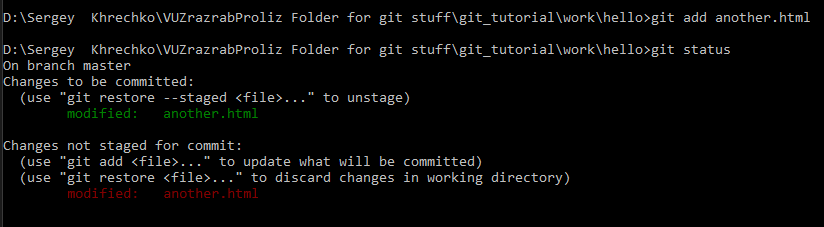


Рисунок 10 – Индексирование и проверка

На рисунке 11 представлено индексирование второго изменения и проверка репозитория.

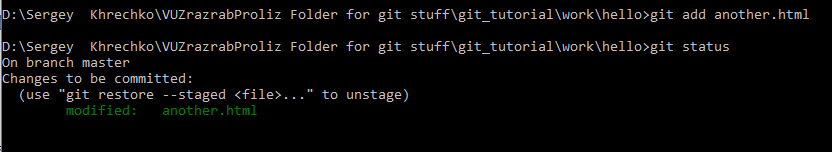


Рисунок 11 – Индексирование и проверка

На рисунке 12 представлено совершение коммита изменений в another.

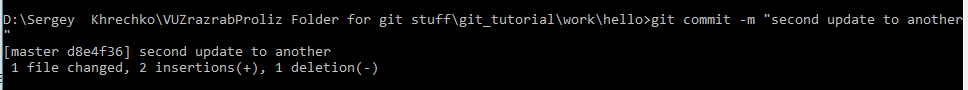


Рисунок 12 – Коммит в another

## Пункт 7

На рисунке 13 представлена история работы с репозиторием.

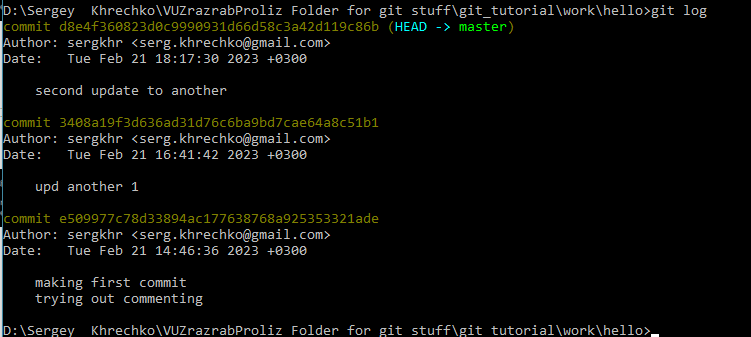


Рисунок 13 – История работы

На рисунке 14 представлены различные отображения истории коммитов. В первом случае это вывод в одну строку, во втором вручную задан формат.

%h – это обозначение для хэша, %ad – для даты, %an – для автора, %s – комментарий, %d - украшения, --date=short – выводит дату в сокращенном виде.

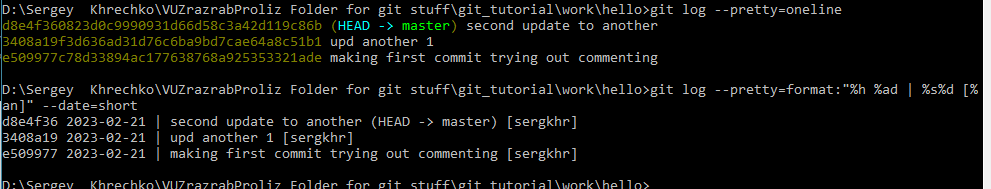


Рисунок 14 – Различные отображения истории

Для удобства был создан псевдоним для красивого вывода истории коммитов. Процесс и результат показаны на рисунке 15.

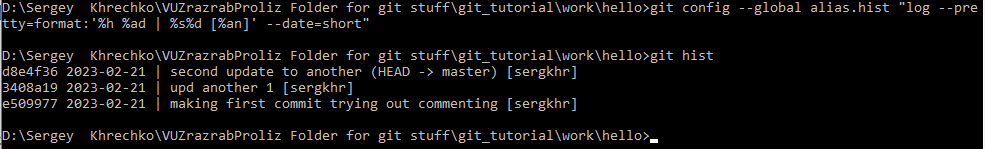


Рисунок 15 – Псевдоним для истории коммитов

## Пункт 8

На рисунке 16 представлено возвращение каталога к состоянию самого первого коммита. Как можно заметить в файле another.html теперь ничего нет, как и должно быть, так как первое наполнение появилось только во втором коммите.

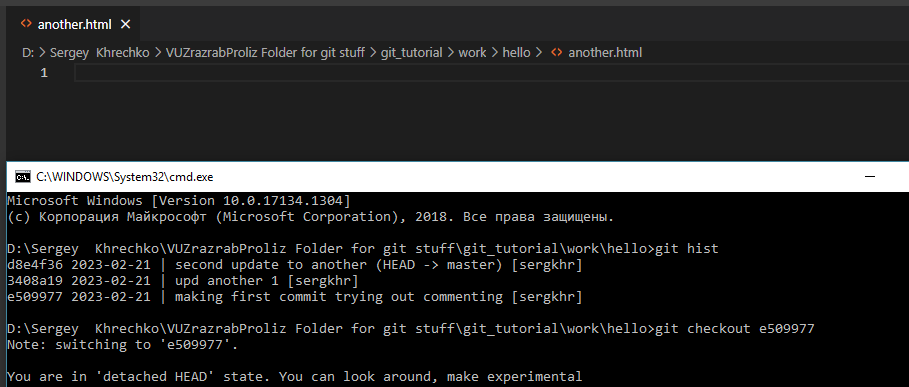


Рисунок 16 – Возвращение к предыдущему состоянию

## Пункт 9

На рисунке 17 представлено создание тегов для коммитов. Версию до изменения мы назвали v1, а последнюю на данный момент версию назвали v2.

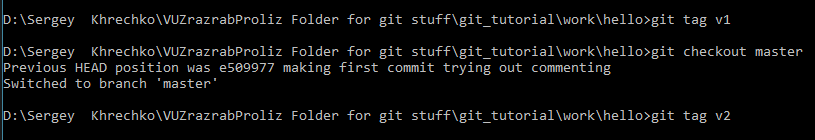


Рисунок 17 – Создание тегов

На рисунке 18 представлена навигация между коммитами с тегами. Нотация v2~1 означает, что мы попадаем в коммит, который на 1 старше коммита v2.

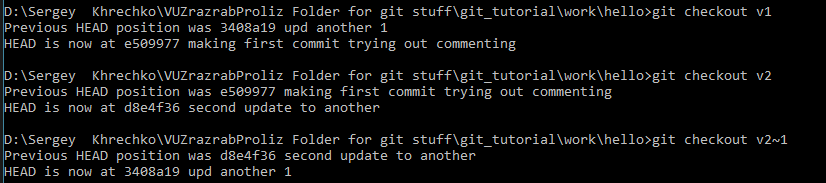


Рисунок 18 – Навигация по тегам

## Пункт 10

На рисунке 19 представлена отмена изменений в рабочем каталоге до их индексирования, при помощи команды checkout.

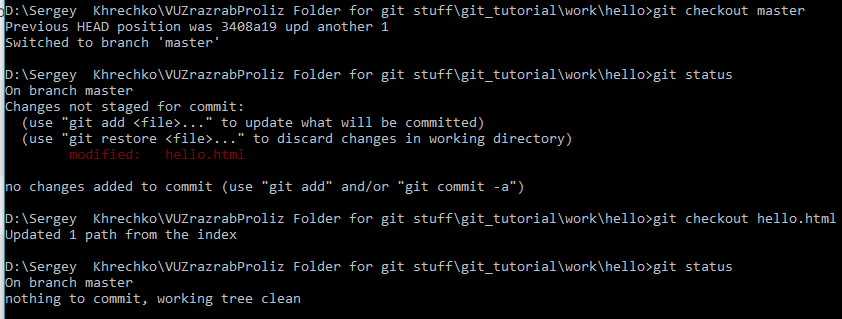


Рисунок 19 – Отмена неиндексированных изменений

На рисунке 20 представлена отмена изменений в рабочем каталоге после индексирования. Команда reset отменяет индексирование, после чего производится отмена уже неиндексированных изменений, как в примере выше.

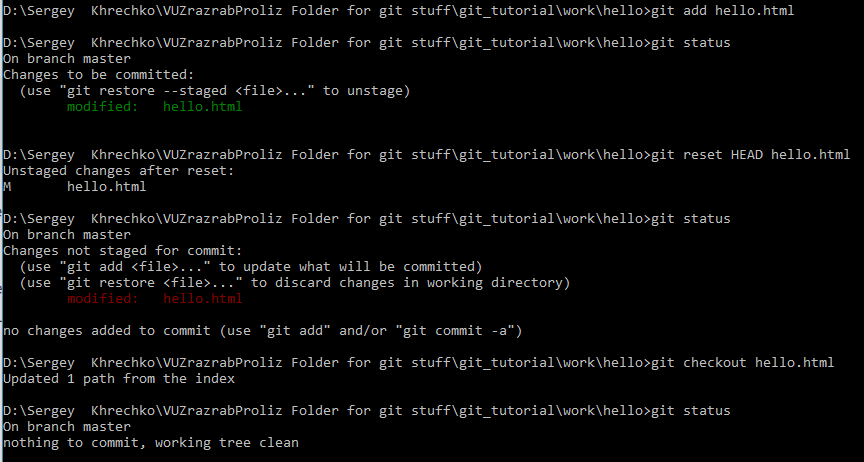


Рисунок 20 – Отмена индексированных изменений

## Пункт 11

На рисунке 21 представлена отмена нежелательного коммита через создание нового, отменяющего изменения, командой revert.

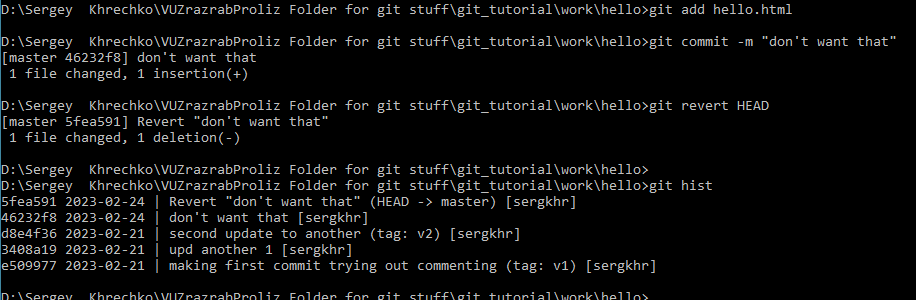


Рисунок 21 – Отмена коммита

На рисунке 22 представлено удаление двух коммитов с помощью команды reset, теперь они не будут отображаться при обычном вызове git log.

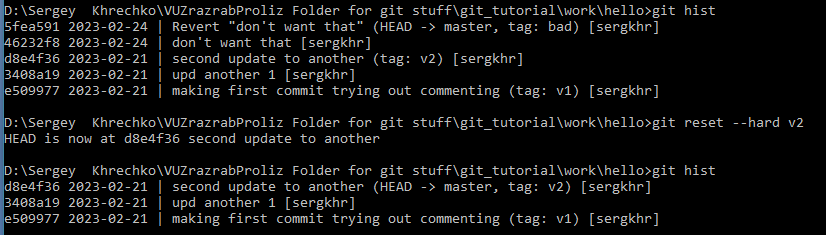


Рисунок 22 – Удаление коммитов

# Часть вторая

## Пункт 1

На рисунке 23 представлен мой аккаунт на github.

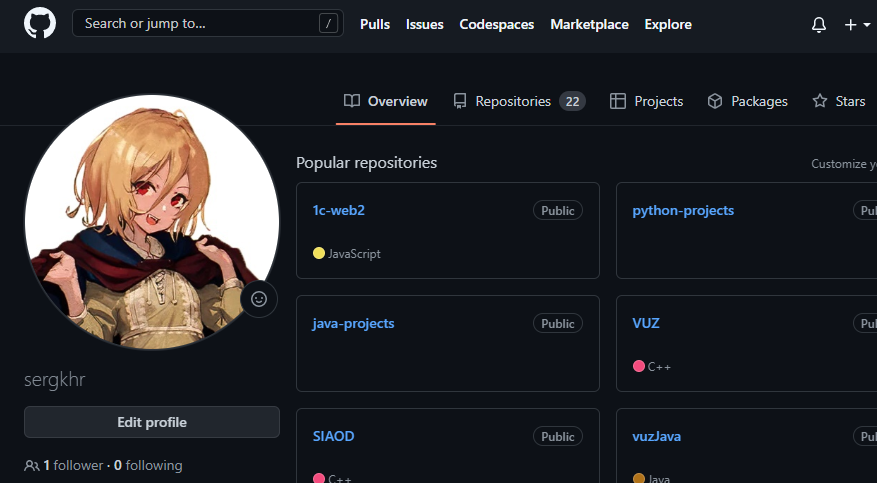


Рисунок 23 – Аккаунт на github

## Пункт 2

На рисунке 24 представлен результат создания репозитория для практики.

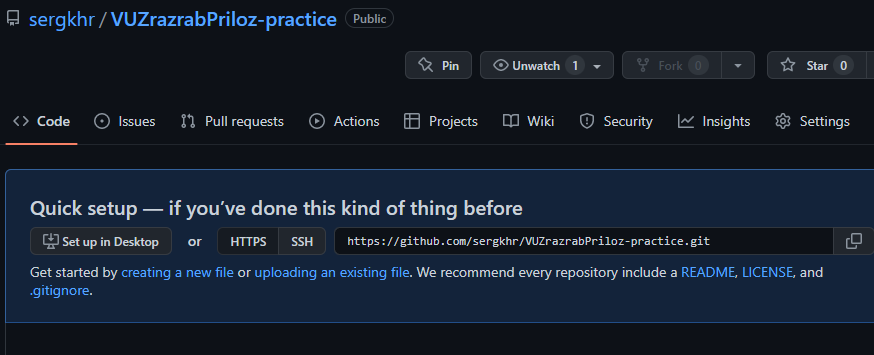


Рисунок 24 – Создан репозиторий на github

На рисунке 25 представлено создание локального репозитория.

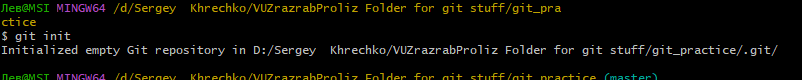


Рисунок 25 – Создание локального репозитория

## Пункт 3

На рисунке 26 представлено создание нескольких файлов в локальном репозитории.



Рисунок 26 – Создание нескольких файлов

На рисунке 27 представлено создание коммита с этими тремя файлами.



Рисунок 27 – Коммит

## Пункт 4

На рисунке 28 представлено получение SSH ключа для настройки соединения с удаленным репозиторием.

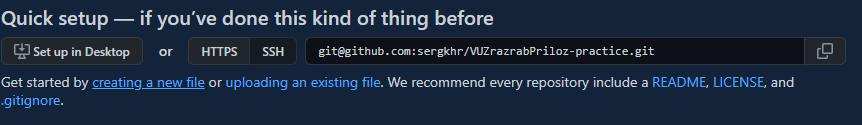


Рисунок 28 – SSH ключ

## Пункт 5

На рисунке 29 представлено связывание локального репозитория с удаленным.



Рисунок 29 – Связывание репозитория

## Пункт 6

На рисунке 30 представлено создание новой ветки и внесение в нее изменений.

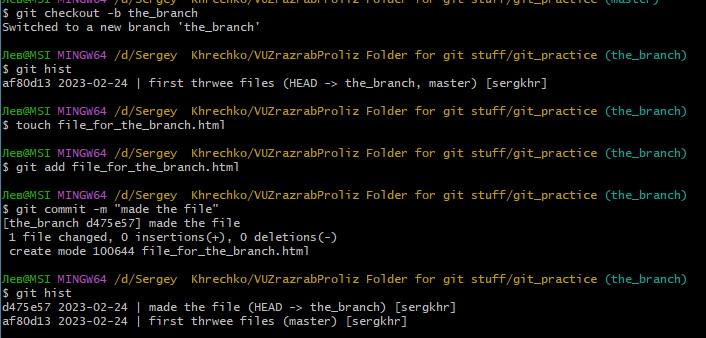


Рисунок 30 – Создание ветки и изменения

На рисунке 31 представлено слияние ветвей the\_branch и master.

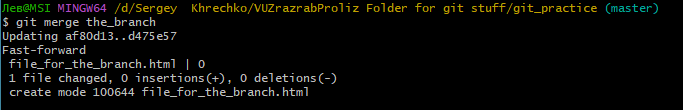


Рисунок 31 – Слияние

## Пункт 7

### Подпункт 1

На рисунке 32 представлено клонирование удаленного репозитория на локальную машину.

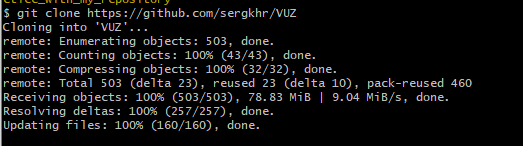


Рисунок 32 – Клонирование репозитория

### Подпункт 2

На рисунке 33 представлено создание новой ветки, а также выведен список всех веток репозитория.

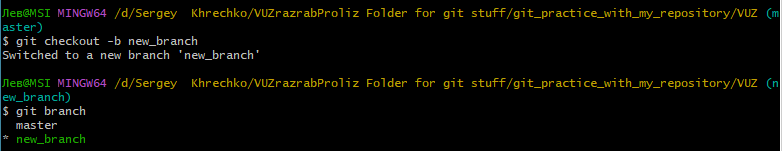


Рисунок 33 – Ветки репозитория

### Подпункт 3

На рисунке 34 представлены записи о трех коммитах на новой ветке. Можно заметить, что до этих коммитов идут коммиты на ветке master.

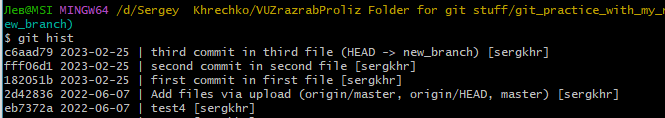


Рисунок 34 – Коммиты на новой ветке

На рисунке 35 представлен третий коммит, предыдущие два сделаны аналогично.

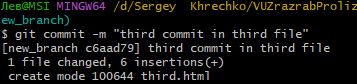


Рисунок 35 – Третий коммит на новой ветке

### Подпункт 4

На рисунке 36 представлена выгрузка изменений в удаленный репозиторий.

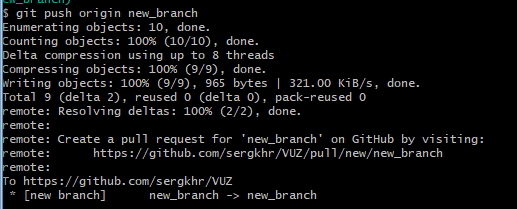


Рисунок 36 – Выгрузка изменений

На рисунке 37 показано, что изменения были успешно выгружены.

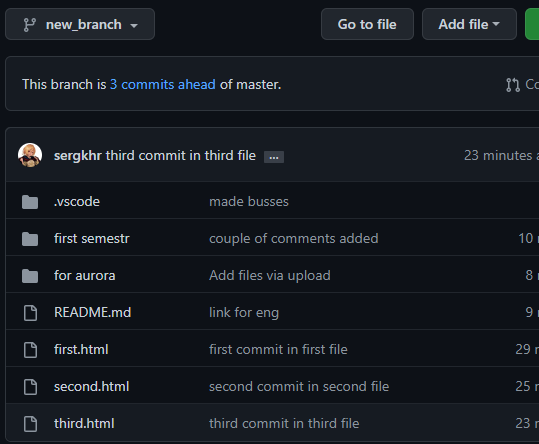


Рисунок 37 – Изменения в репозитории успешны

### Подпункт 5

На рисунке 38 представлено возвращение к состоянию предпоследнего коммита с помощью команды revert.

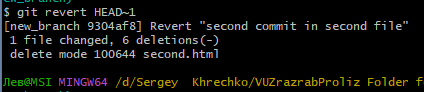


Рисунок 38 – Revert до предпоследнего состояния

### Подпункт 6

На рисунке 39 представлены различия между веткой master и новой веткой.

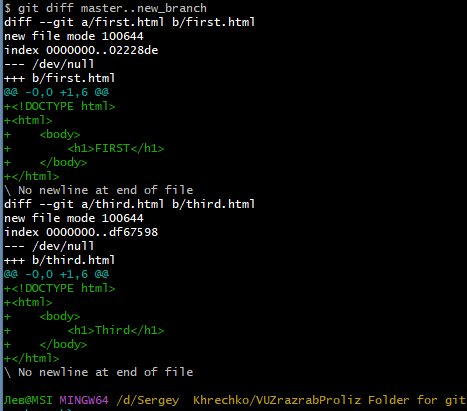


Рисунок 39 – Различия между ветками

### Подпункт 7

На рисунке 40 представлено слияние новой ветки с веткой master.

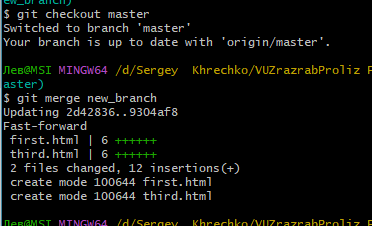


Рисунок 40 – Слияние

# Часть третья

## Пункт 1

На рисунке 41 представлено создание fork’а требуемого по варианту задания репозитория.

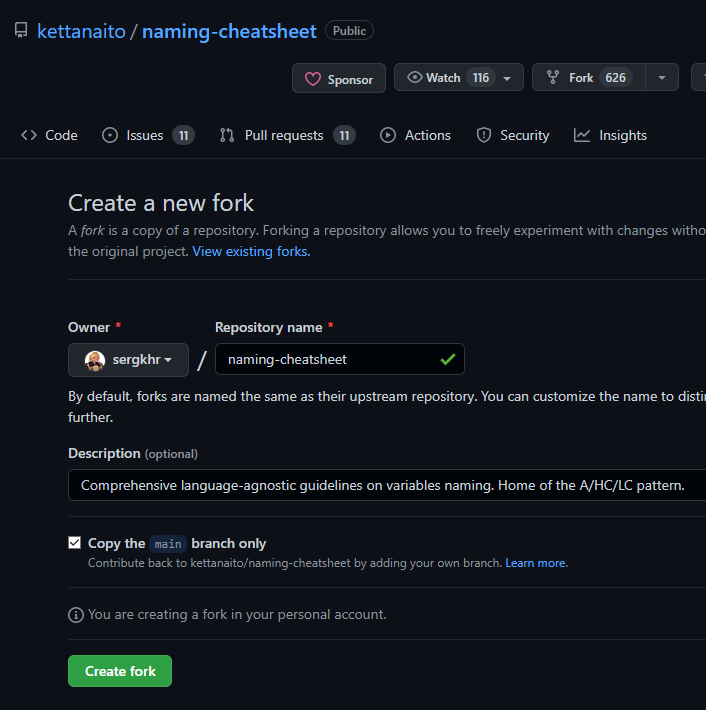


Рисунок 41 – Создание fork’а

## Пункт 2

На рисунке 42 представлено клонирование репозитория на локальную машину.

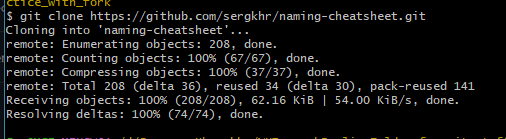


Рисунок 42 – Клонирование репозитория

## Пункт 3

На рисунке 43 представлено создание веток.

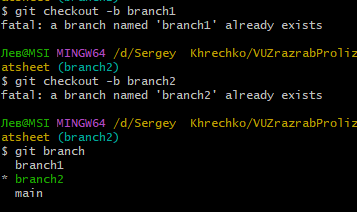


Рисунок 43 – Создание веток

## Пункт 4

На рисунке 44 представлено по три коммита в ветки branch1 и branch2.

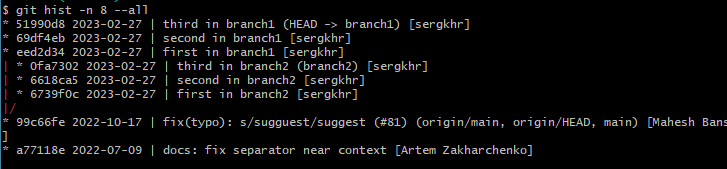


Рисунок 44 – Коммиты

## Пункт 5

На рисунке 45 представлена попытка слияния, результат - конфликт.

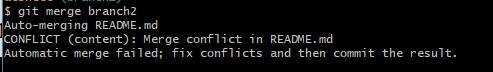


Рисунок 45 – Конфликт при слиянии

На рисунке 46 представлен конфликт.

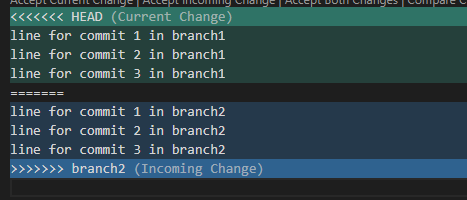


Рисунок 46 – Конфликт

На рисунке 47 представлен коммит с разрешением конфликта.

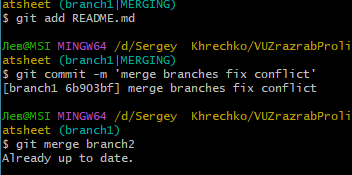


Рисунок 47 – Разрешение конфликта

На рисунке 48 представлена история коммитов. Видно, что слияние с разрешением конфликтов прошло успешно.

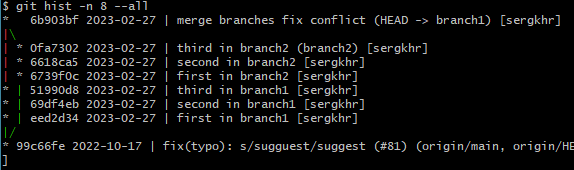


Рисунок 48 – История коммиитов

## Пункт 6

На рисунке 49 представлена выгрузка всех веток в удаленный репозиторий.

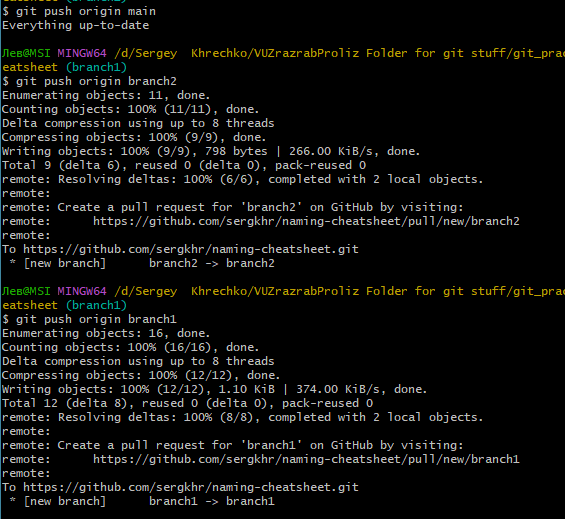


Рисунок 49 – Выгрузка

## Пункт 7

На рисунке 50 представлены еще три коммита в ветку branch1.

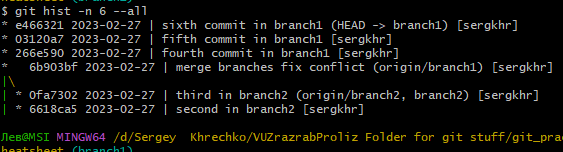


Рисунок 50 – Еще три коммита

## Пункт 8

На рисунке 51 представлено клонирование репозитория в другую папку.

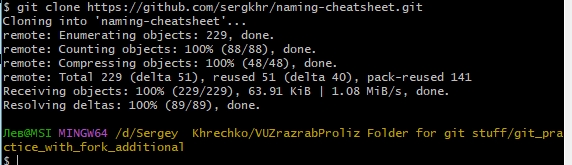


Рисунок 51 – Клонирование

## Пункт 9

На рисунке 52 представлено получение ветки branch1.

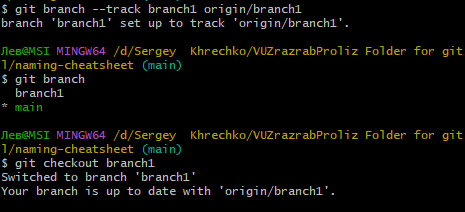


Рисунок 52 – Получение branch1

На рисунке 53 представлены три коммита в ветку branch1 в новой папке. В последнем коммите опечатка в нумерации.

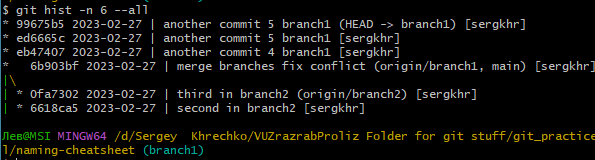


Рисунок 53 – Три коммита

## Пункт 10

На рисунке 54 представлена выгрузка изменений в удаленный репозиторий из новой папки.

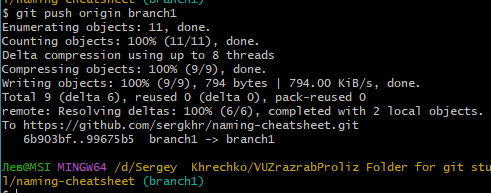


Рисунок 54 – Выгрузка изменений

## Пункт 11

На рисунке 55 представлена выгрузка изменений в удаленный репозиторий из старой папки с опцией --force.

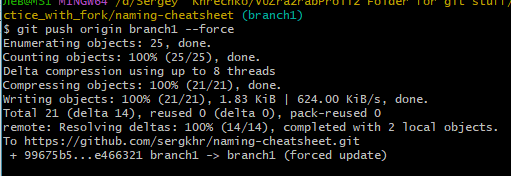


Рисунок 55 – Выгрузка изменений из старой папки

## Пункт 12

На рисунке 56 представлено получение изменений в удаленном репозитории из новой папки. Ожидаемо встречаем конфликт.

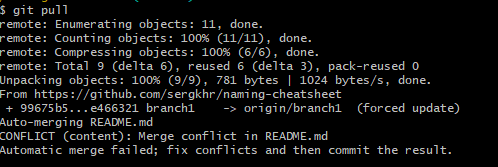


Рисунок 56 – Получение изменений

На рисунке 57 представлен конкретный конфликт.

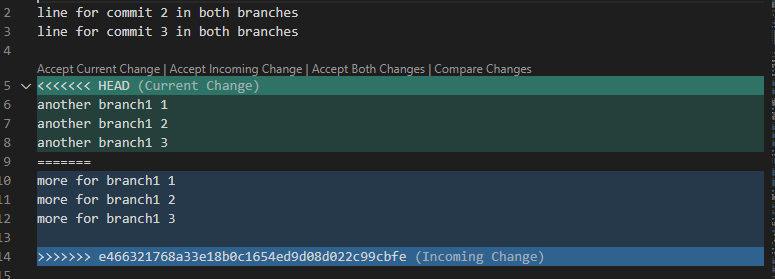


Рисунок 57 –Конфликт

# Ответы на контрольные вопросы

## Вопрос 2

Вопрос:

К какому типу систем контроля версий относится Git?

Ответ:

Git относится к распределенным системам контроля версий. Это означает, что у каждого участника проекта есть полная копия репозитория проекта, а не только его последняя версия, как это было в централизованных системах контроля версий.

## Вопрос 3

Вопрос:

Что такое репозиторий Git?

Ответ:

Репозиторий Git - это место, где хранятся все файлы, данные и история изменений проекта, которые управляются с помощью системы контроля версий Git.

## Вопрос 4

Вопрос:

Что такое коммит?

Ответ:

Коммит в Git - это фиксация изменений в репозитории. Он представляет собой снимок состояния репозитория в определенный момент времени, включая все изменения, которые были внесены в файлы проекта.

## Вопрос 5

Вопрос:

Что такое ветка в репозитории Git?

Ответ:

Ветка (branch) в репозитории Git - это линия разработки, которая представляет собой последовательность коммитов исходящих от определенного коммита, называемого "точкой ветвления".

## Вопрос 10

Вопрос:

Для чего нужен .gitignore?

Ответ:

Чтобы понимать, какие файлы и директории в проекте должны быть проигнорированы при работе с Git. Файлы и директории, указанные в .gitignore, не будут отображаться в списке изменений, не будут добавлены в индекс, и не будут сохранены в коммитах.

## Вопрос 18

Вопрос:

Что сделает команда "git branch" без какого-либо параметра?

Ответ:

Выведет список всех локальных веток в репозитории с пометкой около активной на данный момент.

## Вопрос 27

Вопрос:

Как сделать ветку с названием my\_branch?

Ответ:

Для этого нужно прописать команду: «git branch my\_branch». Это не переведет нас на новую ветку, для этого нужно будет использовать: «git checkout my\_branch», данные два действия можно сократить в одно: «git checkout -b my\_branch».

## Вопрос 28

Вопрос:

Как сделать коммит для ветки my\_branch?

Ответ:

Для этого нужно, чтобы активной веткой была my\_branch и тогда использовать команду: «git commit».

# Выводы

В процессе выполнения работы были освоены основы работы с системой git, а также с GitHub. Была проделана обширная практическая работа для закрепления навыков. Были выполнены поставленные задачи и даны ответы на контрольные вопросы.