МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«МИРЭА – Российский технологический университет»
РТУ МИРЭА

Институт Информационных технологий

Кафедра Математического обеспечения и стандартизации информационных технологий

Отчет по практической работе №1

по дисциплине «Технологии разработки программных приложений»

Тема практической работы: «Системы контроля версий»

Выполнил:
Студент группы ИКБО-20-23 Комисарик М.А.

Проверил: Доцент кафедры МОСИТ,
кандидат технических наук, доцент
Чернов Е.А.

Москва 2025

СОДЕРЖАНИЕ

1 ПОСТАНОВКА ЗАДАЧИ

Цель работы: получить навыки по работе с командной строкой и git’ом.

1.1 Основные команды Git

1. Установить и настроить клиент git на своей рабочей станции.
2. Создать локальный репозиторий и добавьте в него несколько файлов.
3. Внести изменения в один из файлов.
4. Проиндексировать изменения и проверьте состояние.
5. Сделать коммит того, что было проиндексировано в репозиторий. Добавьте к коммиту комментарий.
6. Изменить еще один файл. Добавьте это изменение в индекс git. Изменить файл еще раз. Проверьте состояние и произвести коммит проиндексированного изменения. Теперь добавьте второе изменение в индекс, а затем проверьте состояние с помощью команды git status. Сделать коммит второго изменения.
7. Просмотреть историю коммитов с помощью команды git log. Ознакомиться с параметрами команды и использовать некоторые из них для различного формата отображения истории коммитов.
8. Вернуть рабочий каталог к одному из предыдущих состояний.
9. Изучить, как создавать теги для коммитов для использования в будущем.
10. Отменить некоторые изменения в рабочем каталоге (до и после индексирования).
11. Отменить один из коммитов в локальном репозитории.

1.2 Системы управления репозиториями

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

2. Создать репозиторий на GitHub и на локальной машине, согласно выбранной теме проекта,
3. Создать несколько файлов на локальной машине при помощи консоли,
4. Создать SSH-ключ для авторизации,
5. Связать репозиторий локальной машины с репозиторием на GitHub при помощи консоли,
6. Создать новую ветку в репозитории с помощью команды, произвести в ней какие-нибудь изменения, а после слить с веткой master,
7. Выполнить цепочку действий в репозитории, согласно заданию:

1) Клонировать непустой удаленный репозиторий на локальную машину
2) Создать новую ветку и вывести список всех веток
3) Произвести коммит в ветке master
4) Произвести 3 коммита в новой ветке в разные файлы
5) Выгрузить изменения в удаленный репозиторий
6) Откатить ветку обратно на 2 коммита (в том числе в удаленном репозитории)
7) Вывести в консоли различия между веткой master и новой веткой
8) Перебазировать новую ветку на master

1.3 Работа с ветвлением и оформление кода

1. Сделать форк репозитория по ссылке https://github.com/gto76/python-cheatsheet
2. Склонировать его на локальную машину
3. Создать две ветки branch1 и branch2 от последнего коммита в master'e
4. Провести по 3 коммита в каждую из веток, которые меняют один и тот же кусочек файла
5. Выполнить слияние ветки branch1 в ветку branch2, разрешив конфликты при этом

6. Выгрузить все изменения во всех ветках в удаленный репозиторий
7. Провести еще 3 коммита в ветку branch1
8. Склонировать репозиторий еще раз в другую директорию
9. В новом клоне репозитории сделайте 3 коммита в ветку branch1
10. Выгрузить все изменения из нового репозитория в удаленный репозиторий
11. Вернуться в старый клон с репозиторием, выгрузите изменения с опцией --force
12. Получить все изменения в новом репозитории

2 ВЫПОЛНЕНИЕ ЗАДАНИЯ

2.1 Начало работы с Git

2.1.1 Установка и настройка Git

Для установки Git на Windows требуется скачать установщик с официального сайта (https://git-scm.com/downloads/win).

В процессе установки можно выбрать необходимые опции (например, текстовый редактор коммитов по умолчанию).

Установщик представлен на рисунке 1.

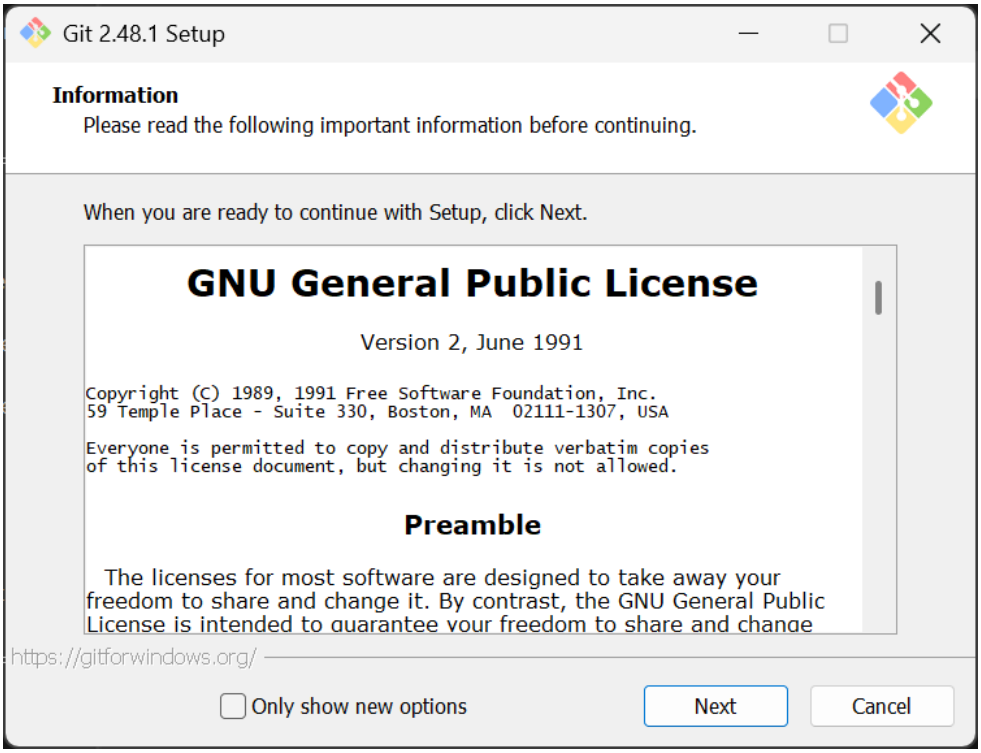

Рисунок 1 – Установка Git

После установки необходимо настроить Git. Процесс настройки представлен на рисунках 2-3.

```
krezo@Krezon MINGW64 ~/Education/SoftDev/Practics/HW1
$ git config --global user.name "Krezon"

krezo@Krezon MINGW64 ~/Education/SoftDev/Practics/HW1
$ git config --global user.email "krezony@gmail.com"

krezo@Krezon MINGW64 ~/Education/SoftDev/Practics/HW1
$ git config --global core.autocrlf true

krezo@Krezon MINGW64 ~/Education/SoftDev/Practics/HW1
$ git config --global core.safecrlf warn

krezo@Krezon MINGW64 ~/Education/SoftDev/Practics/HW1
$ git config --global core.quotepath off
```

Рисунок 2 – Настройка Git, часть 1

```
krezo@Krezon MINGW64 ~/Education/SoftDev/Practics/HW1
$ git config list
diff.astextplain.textconv=astextplain
filter.lfs.clean=git-lfs clean -- %f
filter.lfs.smudge=git-lfs smudge -- %f
filter.lfs.process=git-lfs filter-process
filter.lfs.required=true
http.sslbackend=openssl
http.sslcainfo=C:/Program Files/Git/mingw64/etc/ssl/certs/ca-bundle.crt
core.autocrlf=true
core.fscache=true
core.symlinks=false
pull.rebase=false
credential.helper=manager
credential.https://dev.azure.com.usehttppath=true
init.defaultbranch=master
filter.lfs.clean=git-lfs clean -- %f
filter.lfs.smudge=git-lfs smudge -- %f
filter.lfs.process=git-lfs filter-process
filter.lfs.required=true
user.name=Krezon
user.email=krezony@gmail.com
core.autocrlf=true
core.safecrlf=warn
core.quotepath=off
```

Рисунок 3 – Настройка Git, часть 2

2.1.2 Создание репозитория

Для перехода в другую директорию воспользуемся командой cd <путь к директории> (рисунок 4).

```
$ cd Education/SoftDev/

krezo@Krezon MINGW64 ~/Education/SoftDev
```

Рисунок 4 – Переход в другую папку

Создаем папку, в которой будет лежать наш репозиторий с помощью команды mkdir и переходим в нее с помощью команды cd (рисунок 5).

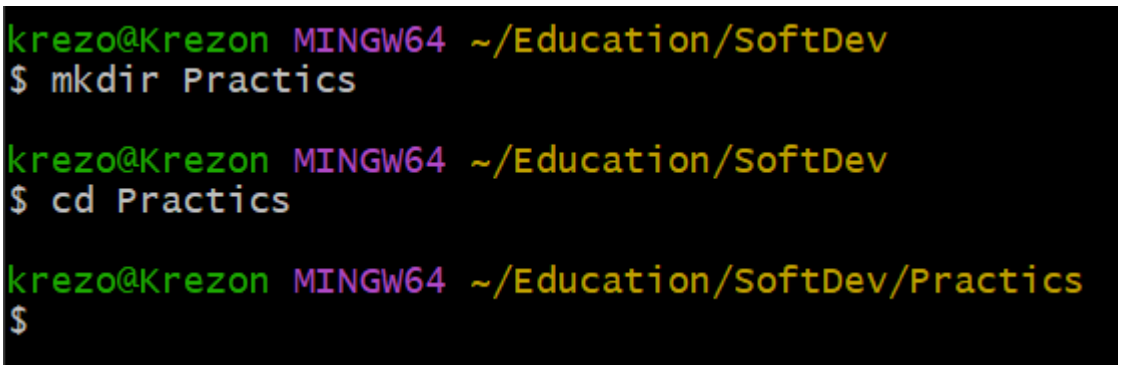

Рисунок 5 – Создание папки и переход в нее

Создадим текстовый файл с помощью текстового редактора vim (рисунки 6-7).

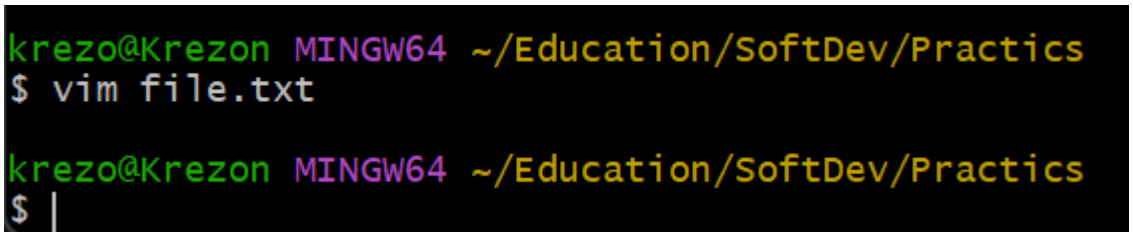

Рисунок 6 – Изменение файла file.txt с помощью текстового редактора vim

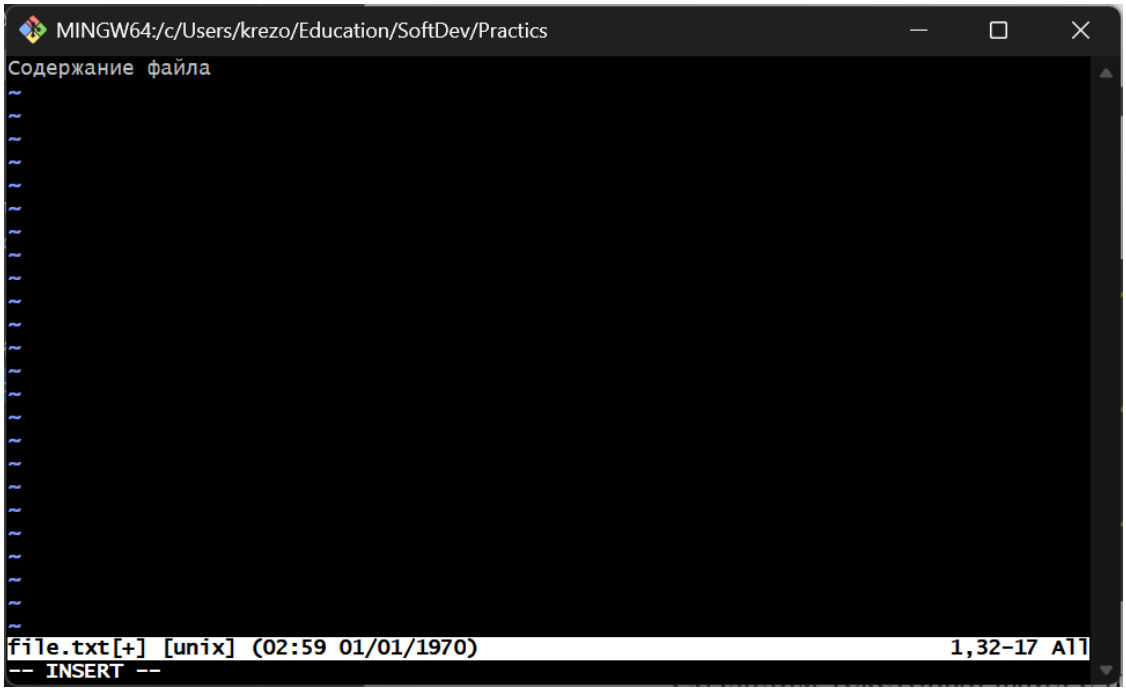

Рисунок 7 – Содержание файла file.txt после изменения

Создадим репозиторий с помощью команды init (рисунок 8).

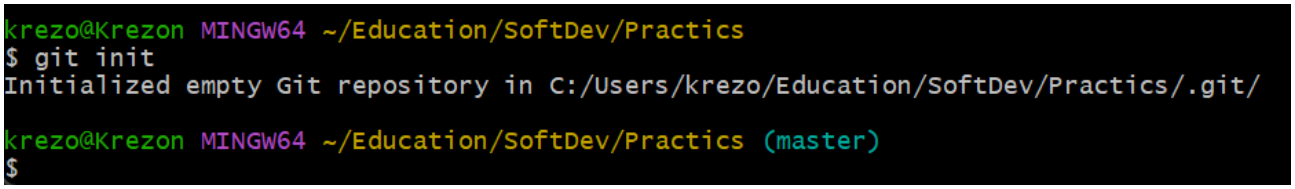

Рисунок 8 – Создание пустого репозитория

2.1.3 Создание коммита

Создадим несколько файлов с помощью команды touch, добавим их в коммит с помощью команды git add и завершим коммит с помощью команды commit (рисунок 9).

```
krezo@Krezon MINGW64 ~/Education/SoftDev/Practics (master)
$ touch file2.txt

krezo@Krezon MINGW64 ~/Education/SoftDev/Practics (master)
$ touch file3.txt

krezo@Krezon MINGW64 ~/Education/SoftDev/Practics (master)
$ git add file.txt
warning: in the working copy of 'file.txt', LF will be replaced by CRLF the next time Git touches i
t

krezo@Krezon MINGW64 ~/Education/SoftDev/Practics (master)
$ git add file2.txt

krezo@Krezon MINGW64 ~/Education/SoftDev/Practics (master)
$ git add file3.txt

krezo@Krezon MINGW64 ~/Education/SoftDev/Practics (master)
$ git commit -m "Первый коммит"
[master (root-commit) 2b8a309] Первый коммит
 3 files changed, 1 insertion(+)
 create mode 100644 file.txt
 create mode 100644 file2.txt
 create mode 100644 file3.txt
```

Рисунок 9 – Создание трех файлов и начальный коммит

2.1.4 Работа с изменениями

Изменим файл file2.txt и закоммитим изменение (рисунки 10-11).

```
krezo@Krezon MINGW64 ~/Education/SoftDev/Practics (master)
$ vim file2.txt

krezo@Krezon MINGW64 ~/Education/SoftDev/Practics (master)
$ git add file2.txt
warning: in the working copy of 'file2.txt', LF will be replaced by CRLF the next time Git touches
it

krezo@Krezon MINGW64 ~/Education/SoftDev/Practics (master)
$ git commit -m "Изменил файл 2"
[master 1d3b731] Изменил файл 2
 1 file changed, 1 insertion(+)
```

Рисунок 10 – Изменение файла file2.txt и коммит изменения

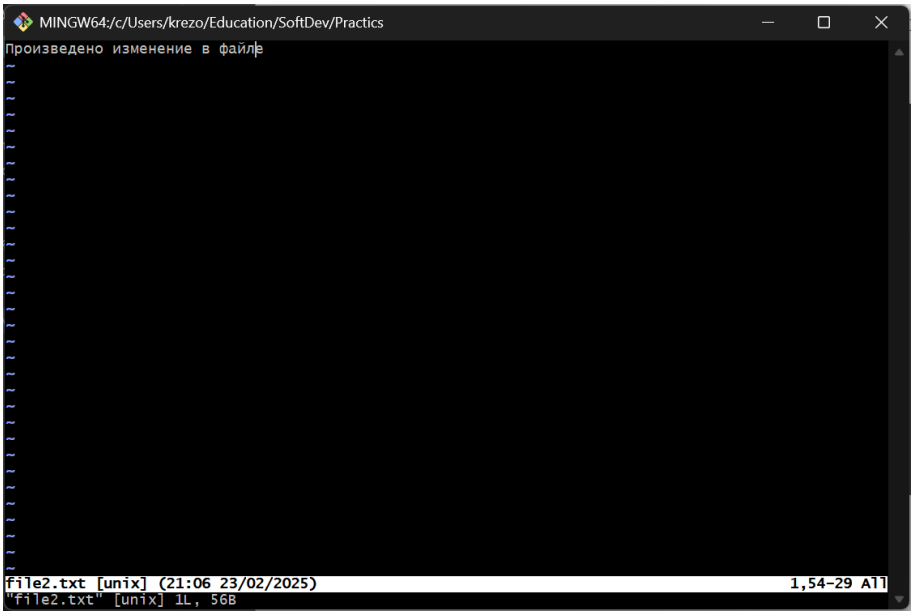

Рисунок 11 – Содержание файла file2.txt после изменения

После этого изменим файл file3.txt и добавим изменение в коммит (рисунки 12-13).

```
krezo@Krezon MINGW64 ~/Education/SoftDev/Practics (master)
$ vim file3.txt

krezo@Krezon MINGW64 ~/Education/SoftDev/Practics (master)
$ git add file3.txt
warning: in the working copy of 'file3.txt', LF will be replaced by CRLF the next time Git touches
it

krezo@Krezon MINGW64 ~/Education/SoftDev/Practics (master)
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   file3.txt
```

Рисунок 12 – Изменение файла file3.txt и индексация изменения

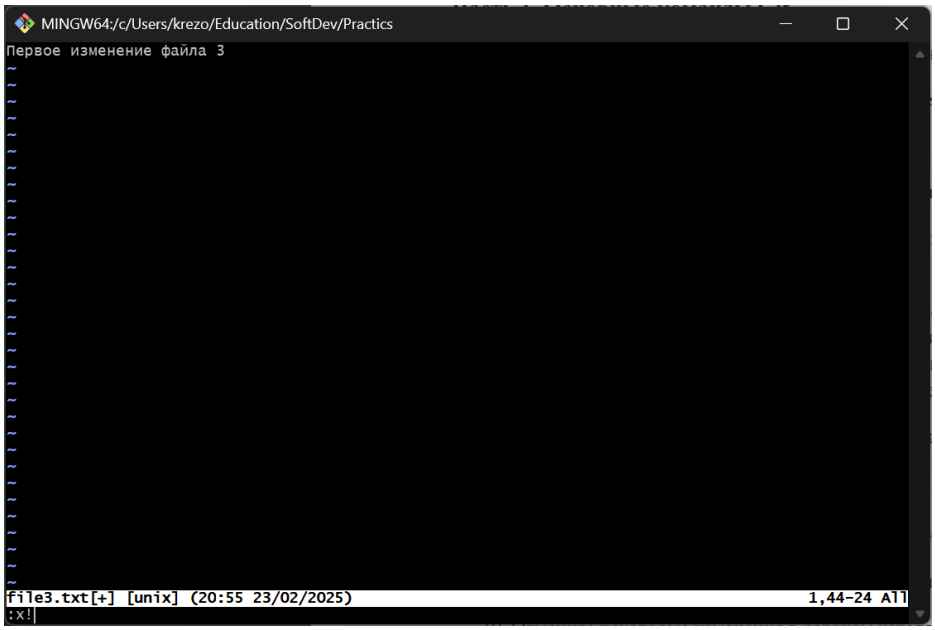

Рисунок 13 – Содержание файла file3.txt после первого изменения

Изменим файл file3.txt еще раз и сделаем коммит без индексирования данного изменения (рисунки 14-15).

```
krezo@Krezon MINGW64 ~/Education/SoftDev/Practics (master)
$ vim file3.txt

krezo@Krezon MINGW64 ~/Education/SoftDev/Practics (master)
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   file3.txt

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   file3.txt


krezo@Krezon MINGW64 ~/Education/SoftDev/Practics (master)
$ git commit -m "Изменение файла 3"
[master 4ab986c] Изменение файла 3
 1 file changed, 1 insertion(+)
```

Рисунок 14 – Изменение файла file3.txt еще раз и коммит

Рисунок 15 – Содержание файла file3.txt после второго изменения

Затем проиндексируем последнее изменение и сделаем коммит (рисунок 16).

```
krezo@Krezon MINGW64 ~/Education/SoftDev/Practics (master)
$ git add file3.txt
warning: in the working copy of 'file3.txt', LF will be replaced by CRLF the next time Git touches
it

krezo@Krezon MINGW64 ~/Education/SoftDev/Practics (master)
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   file3.txt


krezo@Krezon MINGW64 ~/Education/SoftDev/Practics (master)
$ git commit -m "Второе изменение файла 3"
[master bef80da] Второе изменение файла 3
 1 file changed, 1 insertion(+), 1 deletion(-)
```

Рисунок 16 – Индексация изменений и коммит

2.1.5 Просмотр истории коммитов

Посмотрим историю коммитов с помощью команды git log (рисунок 17).

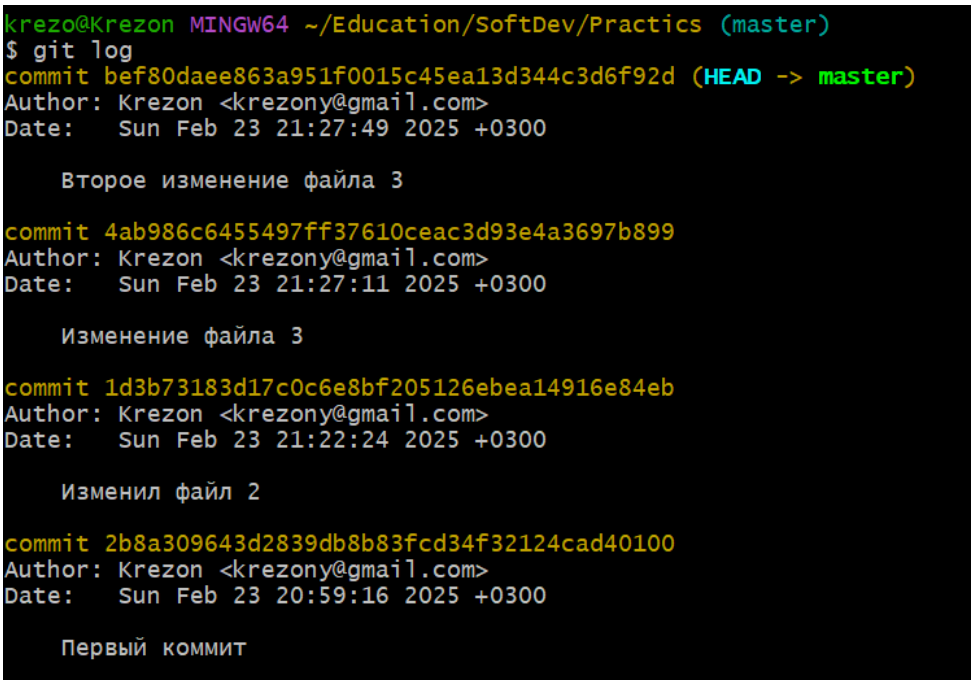

Рисунок 17 – Просмотр истории коммитов с помощью команды git log

С помощью различных опций можно изменять внешний вид команды git log (рисунок 18).

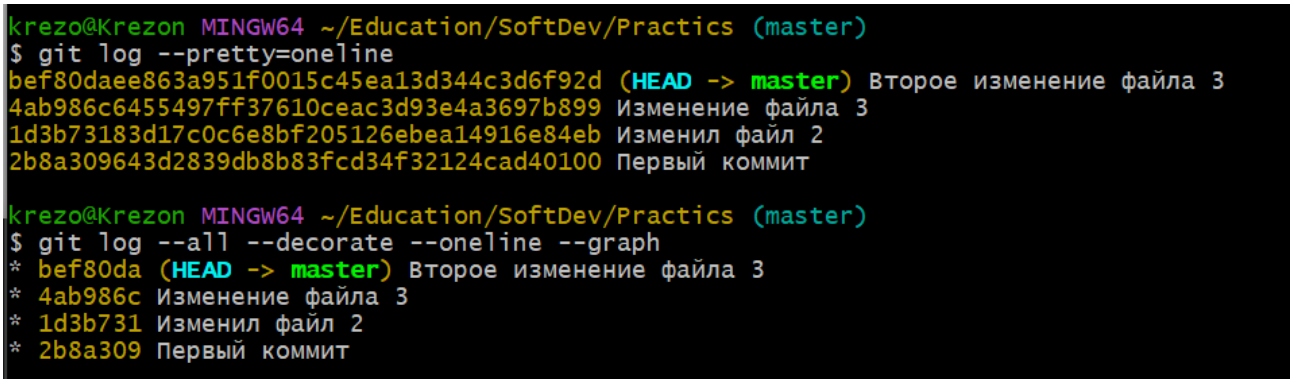

Рисунок 18 – Изменение внешнего вида команды git log

2.1.6 Перемещение между коммитами

Чтобы вернуть рабочий каталог к предыдущему состоянию воспользуемся командой git checkout (рисунок 19).

```
krezo@Krezon MINGW64 ~/Education/SoftDev/Practics (master)
$ git checkout 2b8a309
Note: switching to '2b8a309'.

You are in 'detached HEAD' state. You can look around, make experimental
changes and commit them, and you can discard any commits you make in this
state without impacting any branches by switching back to a branch.

If you want to create a new branch to retain commits you create, you may
do so (now or later) by using -c with the switch command. Example:

  git switch -c <new-branch-name>

Or undo this operation with:

  git switch -

Turn off this advice by setting config variable advice.detachedHead to false

HEAD is now at 2b8a309 Первый коммит
```

Рисунок 19 – Возвращение к предыдущему коммиту с помощью checkout

Проверим, что коммиты действительно откатились (рисунок 20).

```
krezo@Krezon MINGW64 ~/Education/SoftDev/Practics ((2b8a309...))
$ ls
file.txt  file2.txt  file3.txt

krezo@Krezon MINGW64 ~/Education/SoftDev/Practics ((2b8a309...))
$ cat file2.txt

krezo@Krezon MINGW64 ~/Education/SoftDev/Practics ((2b8a309...))
$ git status
HEAD detached at 2b8a309
nothing to commit, working tree clean
```

Рисунок 20 – Проверка отката изменений

Вернемся в текущее состояние репозитория и проверим, что все вернулось (рисунок 21).

```
krezo@Krezon MINGW64 ~/Education/SoftDev/Practics ((2b8a309...))
$ git checkout bef80da
Previous HEAD position was 2b8a309 Первый коммит
HEAD is now at bef80da Второе изменение файла 3

krezo@Krezon MINGW64 ~/Education/SoftDev/Practics ((bef80da...))
$ cat file2.txt
Произведено изменение в файле
```

Рисунок 21 – Проверка возвращения изменений при откате

2.1.7 Использование тегов

Воспользуемся командой **git tag <тег> <хэш коммита>** для создания lightweight тега. Используем команду **git tag** для показа всех существующих тегов (рисунок 22).

```
krezo@Krezon MINGW64 ~/Education/SoftDev/Practics ((bef80da...))
$ git tag 1st 2b8a309

krezo@Krezon MINGW64 ~/Education/SoftDev/Practics ((bef80da...))
$ git tag
1st
```

Рисунок 22 – Создание тега

Для того чтобы узнать информацию о теге можно использовать команду **git show <tag name>** (рисунок 23).

```
krezo@Krezon MINGW64 ~/Education/SoftDev/Practics ((bef80da...))
$ git show 1st
commit 2b8a309643d2839db8b83fcd34f32124cad40100 (tag: 1st)
Author: Krezon <krezony@gmail.com>
Date:   Sun Feb 23 20:59:16 2025 +0300

    Первый коммит

diff --git a/file.txt b/file.txt
new file mode 100644
index 0000000..c0ea46d
--- /dev/null
+++ b/file.txt
@@ -0,0 +1 @@
+Содержание файла
diff --git a/file2.txt b/file2.txt
new file mode 100644
index 0000000..e69de29
diff --git a/file3.txt b/file3.txt
new file mode 100644
index 0000000..e69de29
```

Рисунок 23 – Вывод информации о теге

Для удаления тега можно воспользоваться опцией -d (рисунок 24).

```
krezo@Krezon MINGW64 ~/Education/SoftDev/Practics ((bef80da...))
$ git tag -d 1st
Deleted tag '1st' (was 2b8a309)

krezo@Krezon MINGW64 ~/Education/SoftDev/Practics ((bef80da...))
$ git tag
```

Рисунок 24 – Удаление тега

2.1.8 Отмена изменений

Изменим файл file.txt (рисунки 25-26).

```
krezo@Krezon MINGW64 ~/Education/SoftDev/Practics (master)
$ vim file.txt

krezo@Krezon MINGW64 ~/Education/SoftDev/Practics (master)
$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   file.txt

no changes added to commit (use "git add" and/or "git commit -a")
```

Рисунок 25 – Изменение файла file.txt

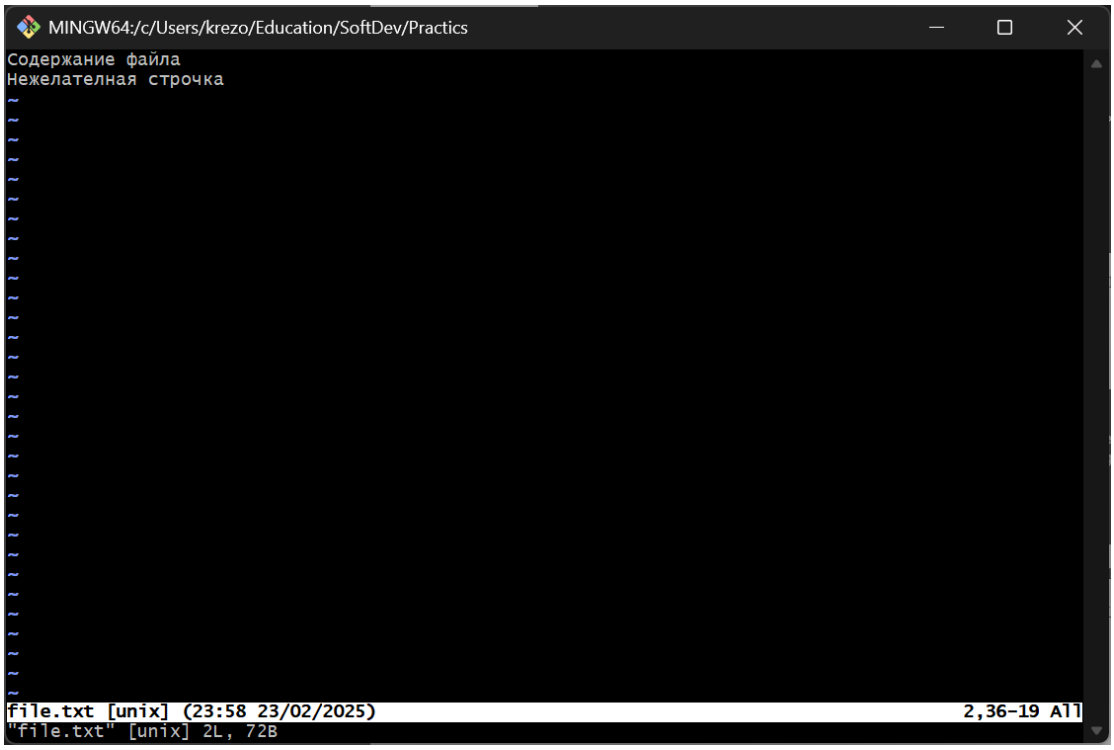

Рисунок 26 – Содержание файла file.txt

Для отмены изменений до индексирования можно воспользоваться командой **git restore <путь>** (рисунок 27).

```
krezo@Krezon MINGW64 ~/Education/SoftDev/Practics (master)
$ git restore file.txt

krezo@Krezon MINGW64 ~/Education/SoftDev/Practics (master)
$ git status
On branch master
nothing to commit, working tree clean

krezo@Krezon MINGW64 ~/Education/SoftDev/Practics (master)
$ cat file.txt
Содержание файла
```

Рисунок 27 – Отмена изменений до индексации

Также, чтобы отменить сразу все изменения можно воспользоваться командой “**git restore .**”.

Изменим файл file.txt еще раз (рисунок 28).

```
krezo@Krezon MINGW64 ~/Education/SoftDev/Practics (master)
$ vim file.txt

krezo@Krezon MINGW64 ~/Education/SoftDev/Practics (master)
$ cat file.txt
Содержание файла
Второй нежелательный комментарий

krezo@Krezon MINGW64 ~/Education/SoftDev/Practics (master)
$ git add file.txt

krezo@Krezon MINGW64 ~/Education/SoftDev/Practics (master)
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   file.txt
```

Рисунок 28 – Изменение файла file.txt еще раз

Для отмены индексированных изменений нужно воспользоваться командой **git restore --staged <путь>** или **git reset [<ветка>] <путь>**, а затем **git restore [--source=<ветка>] <путь>**, так как **git reset** не возвращает изменения, а лишь убирает файл из индексированных (рисунок 29).

```
krezo@Krezon MINGW64 ~/Education/SoftDev/Practics (master)
$ git reset file.txt
Unstaged changes after reset:
M       file.txt

krezo@Krezon MINGW64 ~/Education/SoftDev/Practics (master)
$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   file.txt

no changes added to commit (use "git add" and/or "git commit -a")

krezo@Krezon MINGW64 ~/Education/SoftDev/Practics (master)
$ git restore file.txt

krezo@Krezon MINGW64 ~/Education/SoftDev/Practics (master)
$ cat file.txt
Содержание файла

krezo@Krezon MINGW64 ~/Education/SoftDev/Practics (master)
$ git status
On branch master
nothing to commit, working tree clean
```

Рисунок 29 – Отмена индексированных изменений

2.1.9 Отмена коммита

Изменим файлы file2.txt и file3.txt (рисунок 30).

```
krezo@Krezon MINGW64 ~/Education/SoftDev/Practics (master)
$ vim file2.txt

krezo@Krezon MINGW64 ~/Education/SoftDev/Practics (master)
$ vim file3.txt

krezo@Krezon MINGW64 ~/Education/SoftDev/Practics (master)
$ cat file2.txt
Произведено изменение в файле
Добавлено нежелательное изменение в файле

krezo@Krezon MINGW64 ~/Education/SoftDev/Practics (master)
$ cat file3.txt
Второе изменение файла 3
Нежелательное изменение
```

Рисунок 30 – Изменение файлов file2.txt и file3.txt

Добавим изменения и сделаем коммит (рисунок 31).

```
krezo@Krezon MINGW64 ~/Education/SoftDev/Practics (master)
$ git add .

krezo@Krezon MINGW64 ~/Education/SoftDev/Practics (master)
$ git commit -m "Нежелательный коммит"
[master 9b44e8c] Нежелательный коммит
 2 files changed, 2 insertions(+)
```

Рисунок 31 – Создание коммита

Для отмены коммита можно использовать команду **git revert <коммит>**, которая сделает новый коммит, отменяющий каждое действие выбранного коммита (рисунок 32).

```
[master d778a61] Revert "Нежелательный коммит"
 2 files changed, 2 deletions(-)

krezo@Krezon MINGW64 ~/Education/SoftDev/Practics (master)
$ git log
commit d778a6153d9bcc652c9cd08e754dbcc8dd914510 (HEAD -> master)
Author: Krezon <krezony@gmail.com>
Date:   Mon Feb 24 01:28:38 2025 +0300

    Revert "Нежелательный коммит"

    This reverts commit 9b44e8c836a6298547400c28936c96c9cc4602d1.

    Отмена нежелательного коммита.
```

Рисунок 32 – Отмена коммита с помощью команды revert

2.2 Системы управления репозиториями

2.2.1 Создание GitHub репозитория

Для создания репозитория на GitHub нужно воспользоваться официальным вебсайтом, прежде зарегистрировавшись (рисунок 33).

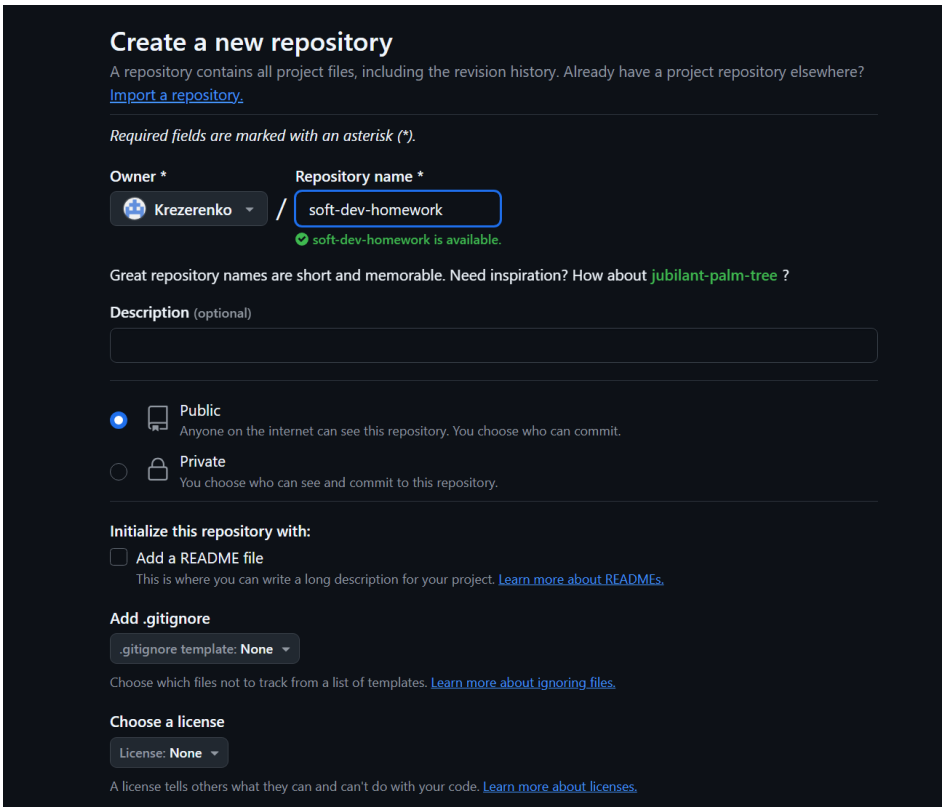

Рисунок 33 – Создание нового репозитория на GitHub

Далее создадим SSH-ключ для авторизации (рисунок 34).

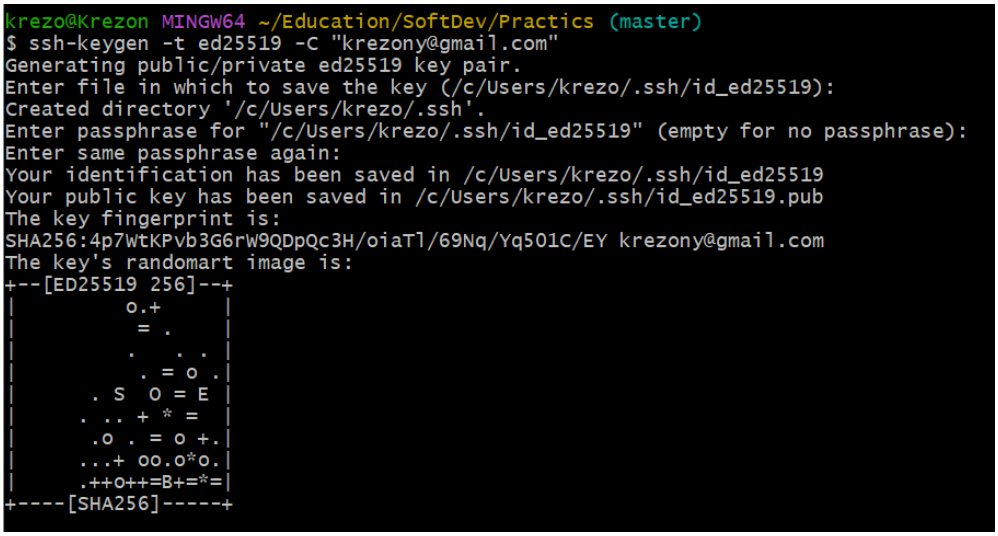

Рисунок 34 – Создание нового SSH ключа

Проверим ключ и добавим его в ssh-agent (рисунок 35).

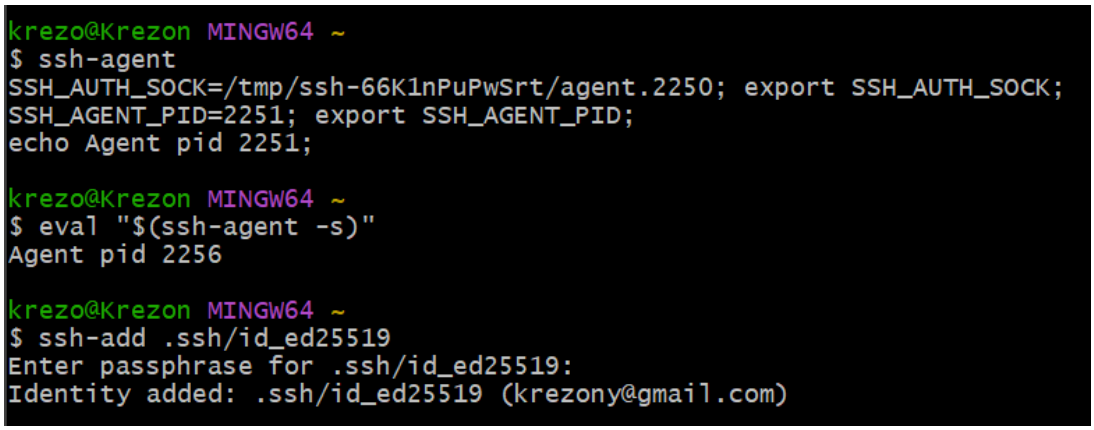

Рисунок 35 – Добавление нового SSH ключа в SSH-Agent

Добавим ключ в аккаунт GitHub (рисунок 36).

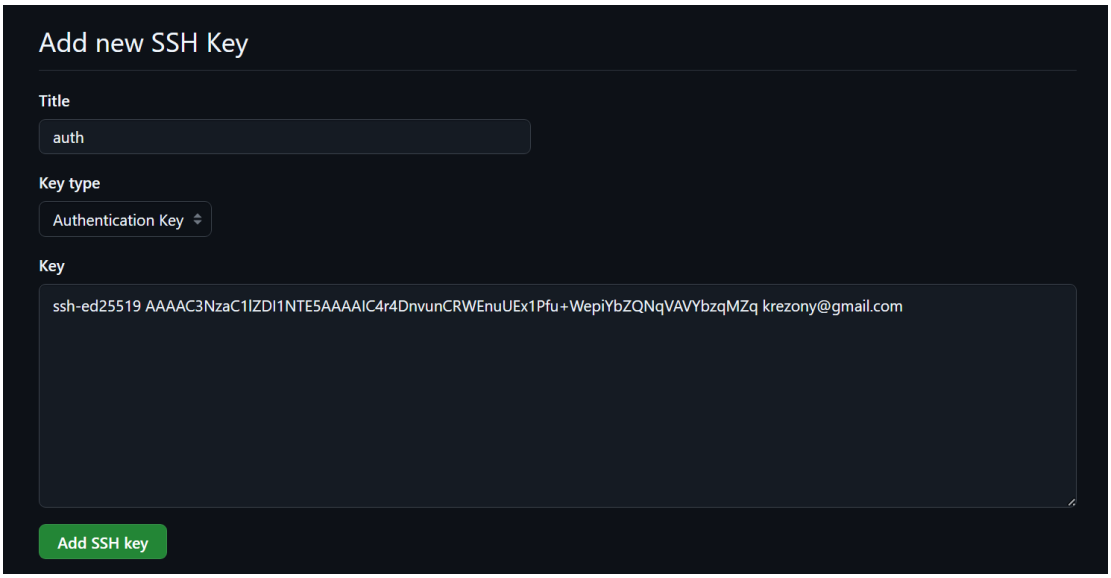

Рисунок 36 – Добавление нового ключа SSH в аккаунт GitHub

2.2.2 Создание локального репозитория

Заполним несколькими файлами пустую папку, добавим их в новый репозиторий, а затем свяжем с удаленным репозиторием (рисунок 37).

```
krezo@Krezon MINGW64 ~/Education/SoftDev/Practics/HW1_2
$ vim readme.md

krezo@Krezon MINGW64 ~/Education/SoftDev/Practics/HW1_2
$ vim code.c

krezo@Krezon MINGW64 ~/Education/SoftDev/Practics/HW1_2
$ git init
Initialized empty Git repository in C:/Users/krezo/Education/SoftDev/Practics/HW
1_2/.git/

krezo@Krezon MINGW64 ~/Education/SoftDev/Practics/HW1_2 (master)
$ git remote add origin git@github.com:Krezerenko/soft-dev-homework.git

krezo@Krezon MINGW64 ~/Education/SoftDev/Practics/HW1_2 (master)
$ git add --all
warning: in the working copy of 'code.c', LF will be replaced by CRLF the next t
ime Git touches it
warning: in the working copy of 'readme.md', LF will be replaced by CRLF the nex
t time Git touches it

krezo@Krezon MINGW64 ~/Education/SoftDev/Practics/HW1_2 (master)
$ git commit -m "Первый коммит"
[master (root-commit) 468db14] Первый коммит
 2 files changed, 2 insertions(+)
 create mode 100644 code.c
 create mode 100644 readme.md
```

Рисунок 37 – Заполнение пустой папки и связка с удаленным репозиторием

2.2.3 Работа с ветками

Создадим новую ветку **second** и произведем в ней некоторые изменения, а затем сольем с веткой **master** (рисунок 38).

```
krezo@Krezon MINGW64 ~/Education/SoftDev/Practics/HW1_2 (master)
$ git checkout -b second
Switched to a new branch 'second'

krezo@Krezon MINGW64 ~/Education/SoftDev/Practics/HW1_2 (second)
$ vim code.c

krezo@Krezon MINGW64 ~/Education/SoftDev/Practics/HW1_2 (second)
$ git add code.c
warning: in the working copy of 'code.c', LF will be replaced by CRLF the next time Git touches
it

krezo@Krezon MINGW64 ~/Education/SoftDev/Practics/HW1_2 (second)
$ git commit -m "Добавил новый код"
[second 75b8f45] Добавил новый код
 1 file changed, 1 insertion(+)

krezo@Krezon MINGW64 ~/Education/SoftDev/Practics/HW1_2 (second)
$ git checkout master
Switched to branch 'master'

krezo@Krezon MINGW64 ~/Education/SoftDev/Practics/HW1_2 (master)
$ git merge second
Updating 468db14..75b8f45
Fast-forward
 code.c | 1 +
 1 file changed, 1 insertion(+)
```

Рисунок 38 – Создание новой ветки second и слияние с master

2.2.4 Задание согласно варианту

Склонируем репозиторий в новую папку (рисунок 39).

```
krezo@Krezon MINGW64 ~/Education/SoftDev/Practics/HW1_2.2
$ git clone git@github.com:Krezerenko/unity-manual.git .
Cloning into '.'...
remote: Enumerating objects: 134, done.
remote: Counting objects: 100% (134/134), done.
remote: Compressing objects: 100% (89/89), done.
remote: Total 134 (delta 68), reused 105 (delta 39), pack-reused 0 (from 0)
Receiving objects: 100% (134/134), 1.31 MiB | 2.68 MiB/s, done.
Resolving deltas: 100% (68/68), done.
```

Рисунок 39 – Клонирование репозитория в новую папку

Создадим новую ветку и выведем список новых веток (рисунок 40).

```
krezo@Krezon MINGW64 ~/Education/SoftDev/Practics/HW1_2.2 (main)
$ git checkout -b newbranch
Switched to a new branch 'newbranch'

krezo@Krezon MINGW64 ~/Education/SoftDev/Practics/HW1_2.2 (newbranch)
$ git branch
  main
* newbranch
```

Рисунок 40 – Создание новой ветки и вывод списка всех веток

Сделаем коммит в ветке main (рисунок 41).

```
krezo@Krezon MINGW64 ~/Education/SoftDev/Practics/HW1_2.2 (newbranch)
$ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

krezo@Krezon MINGW64 ~/Education/SoftDev/Practics/HW1_2.2 (main)
$ vim readme.md

krezo@Krezon MINGW64 ~/Education/SoftDev/Practics/HW1_2.2 (main)
$ git add readme.md
warning: in the working copy of 'readme.md', LF will be replaced by CRLF the next time Git touch
es it

krezo@Krezon MINGW64 ~/Education/SoftDev/Practics/HW1_2.2 (main)
$ git commit -m "Добавил readme файл"
[main 6c52dac] Добавил readme файл
 1 file changed, 1 insertion(+)
 create mode 100644 readme.md
```

Рисунок 41 – Коммит в ветке main

Перейдем на ветку newbranch и сделаем на ней 3 коммита (рисунки 42-43).

```
krezo@Krezon MINGW64 ~/Education/SoftDev/Practics/HW1_2.2 (main)
$ git checkout newbranch
Switched to branch 'newbranch'

krezo@Krezon MINGW64 ~/Education/SoftDev/Practics/HW1_2.2 (newbranch)
$ vim js/quiz.js

krezo@Krezon MINGW64 ~/Education/SoftDev/Practics/HW1_2.2 (newbranch)
$ git add js/quiz.js

krezo@Krezon MINGW64 ~/Education/SoftDev/Practics/HW1_2.2 (newbranch)
$ git commit -m "Добавил метод"
[newbranch 4b2de4f] Добавил метод
 1 file changed, 6 insertions(+), 1 deletion(-)
```

Рисунок 42 – Коммит в ветку newbranch

```
krezo@Krezon MINGW64 ~/Education/SoftDev/Practics/HW1_2.2 (newbranch)
$ vim js/guide-nav.js

krezo@Krezon MINGW64 ~/Education/SoftDev/Practics/HW1_2.2 (newbranch)
$ git add js/guide-nav.js

krezo@Krezon MINGW64 ~/Education/SoftDev/Practics/HW1_2.2 (newbranch)
$ git commit -m "Добавил комментарий для файла навигации"
[newbranch fd543a7] Добавил комментарий для файла навигации
 1 file changed, 3 insertions(+), 1 deletion(-)

krezo@Krezon MINGW64 ~/Education/SoftDev/Practics/HW1_2.2 (newbranch)
$ vim css/nav.css

[1]+  Stopped                 vim css/nav.css

krezo@Krezon MINGW64 ~/Education/SoftDev/Practics/HW1_2.2 (newbranch)
$vim css/nav.css

krezo@Krezon MINGW64 ~/Education/SoftDev/Practics/HW1_2.2 (newbranch)
$ git add css/nav.css

krezo@Krezon MINGW64 ~/Education/SoftDev/Practics/HW1_2.2 (newbranch)
$ git commit -m "Убрал ghost код из файла nav.css"
[newbranch e6b98e9] Убрал ghost код из файла nav.css
 1 file changed, 1 insertion(+), 9 deletions(-)
```

Рисунок 43 – Еще два коммита в ветку newbranch

Выгрузим изменения в удаленный репозиторий (рисунок 44).

```
krezo@Krezon MINGW64 ~/Education/SoftDev/Practics/HW1_2.2 (newbranch)
$ git push origin --all
Enumerating objects: 21, done.
Counting objects: 100% (21/21), done.
Delta compression using up to 20 threads
Compressing objects: 100% (15/15), done.
Writing objects: 100% (15/15), 1.62 KiB | 831.00 KiB/s, done.
Total 15 (delta 10), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (10/10), completed with 5 local objects.
To github.com:Krezerenko/unity-manual.git
   a8b5aed..6c52dac  main -> main
 * [new branch]      newbranch -> newbranch
```

Рисунок 44 – Выгрузка изменений в удаленный репозиторий

Отменим последние 2 коммита и выгрузим изменения в удаленный репозиторий (рисунок 45).

```
krezo@Krezon MINGW64 ~/Education/SoftDev/Practics/HW1_2.2 (newbranch)
$ git revert --no-commit HEAD~2..HEAD

krezo@Krezon MINGW64 ~/Education/SoftDev/Practics/HW1_2.2 (newbranch|REVERTING)
$ git commit -m "Откат двух последних коммитов"
[newbranch 936f417] Откат двух последних коммитов
 2 files changed, 10 insertions(+), 4 deletions(-)

krezo@Krezon MINGW64 ~/Education/SoftDev/Practics/HW1_2.2 (newbranch)
$ git push origin --all
Enumerating objects: 11, done.
Counting objects: 100% (11/11), done.
Delta compression using up to 20 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (6/6), 589 bytes | 589.00 KiB/s, done.
Total 6 (delta 5), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (5/5), completed with 5 local objects.
To github.com:Krezerenko/unity-manual.git
   e6b98e9..936f417  newbranch -> newbranch
```

Рисунок 45 – Отмена последних двух коммитов и выгрузка изменений

Выведем различия между двумя ветками (рисунок 46).

```
krezo@Krezon MINGW64 ~/Education/SoftDev/Practics/HW1_2.2 (newbranch)
$ git diff main newbranch
diff --git a/js/quiz.js b/js/quiz.js
index a3db7df..47acf87 100644
--- a/js/quiz.js
+++ b/js/quiz.js
@@ -60,4 +60,9 @@ function onAnswerCheck(event)
             label.classList.remove("wrong");
         }
     }
-}
\ No newline at end of file
+}
+
+function someNewFunction1()
+{
+    doSomething();
+}
diff --git a/readme.md b/readme.md
deleted file mode 100644
index 840a58a..0000000
--- a/readme.md
+++ /dev/null
@@ -1 +0,0 @@
-# Это сайт для курсовой работы по предмету Проектирование и Разработка Клиентских Частей Интернет-Ресурсов.
```

Рисунок 46 – Вывод различий между двумя ветками

Перебазируем новую ветку на ветку **main** (рисунки 47-48).

```
krezo@Krezon MINGW64 ~/Education/SoftDev/Practics/HW1_2.2 (newbranch)
$ git rebase main
Successfully rebased and updated refs/heads/newbranch.
```

Рисунок 47 – Перебазирование новой ветки на ветку main

```
krezo@Krezon MINGW64 ~/Education/SoftDev/Practics/HW1_2.2 (newbranch)
$ git lg
* 56376c3 (HEAD -> newbranch) Откат двух последних коммитов
* 579cdaf Убрал ghost код из файла nav.css
* 2481f0f Добавил комментарий для файла навигации
* 1088044 Добавил метод
* 6c52dac (origin/main, origin/HEAD, main) Добавил readme файл
| * 936f417 (origin/newbranch) Откат двух последних коммитов
| * e6b98e9 Убрал ghost код из файла nav.css
| * fd543a7 Добавил комментарий для файла навигации
| * 4b2de4f Добавил метод
|/
* a8b5aed Убрал устаревшие иконки вебсайта.
```

Рисунок 48 – История изменений в процессе выполнения задания 2

2.3 Работа с ветвлением и оформление кода

Вариант: https://github.com/gto76/python-cheatsheet.

Сделаем форк этого репозитория (рисунок 49).

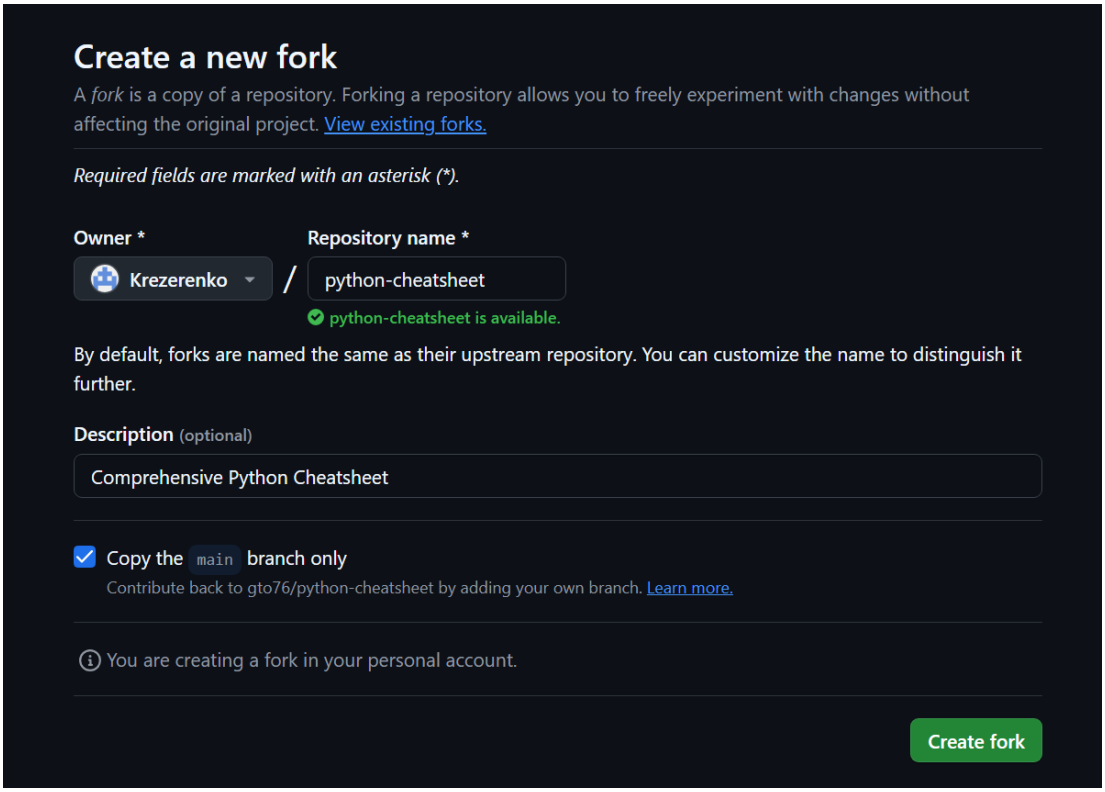

Рисунок 49 – Создание форка репозитория по заданию

Склонируем этот репозиторий в пустую папку (рисунок 50).

```
krezo@Krezon MINGW64 ~/Education/SoftDev/Practics/HW1_3
$ git clone git@github.com:Krezerenko/python-cheatsheet.git ./
Cloning into '.'...
remote: Enumerating objects: 10237, done.
remote: Counting objects: 100% (124/124), done.
remote: Compressing objects: 100% (68/68), done.
remote: Total 10237 (delta 73), reused 56 (delta 56), pack-reused 10113 (from 3)
Receiving objects: 100% (10237/10237), 12.12 MiB | 4.20 MiB/s, done.
Resolving deltas: 100% (6083/6083), done.
```

Рисунок 50 – Клонирование репозитория по варианту в новую папку

Создадим две новые ветки (рисунок 51).

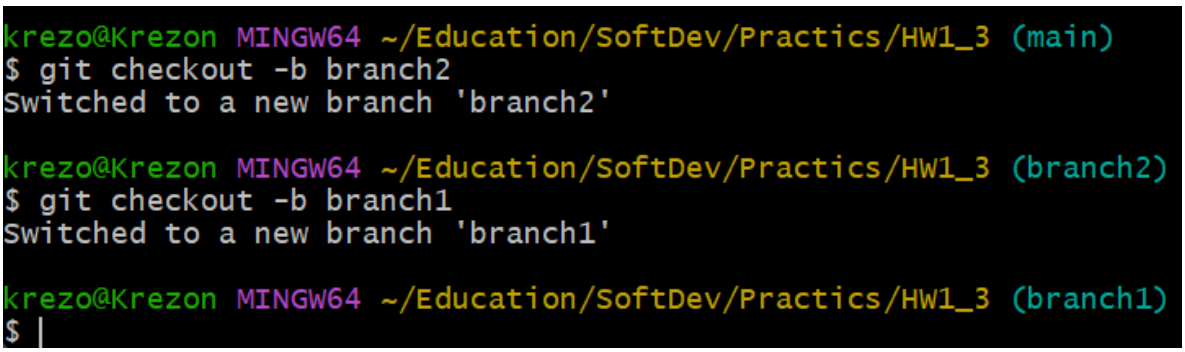

Рисунок 51 – Создание веток branch1 и branch2 в клонированном репозитории

Сделаем три коммита в первую ветку (рисунок 52).

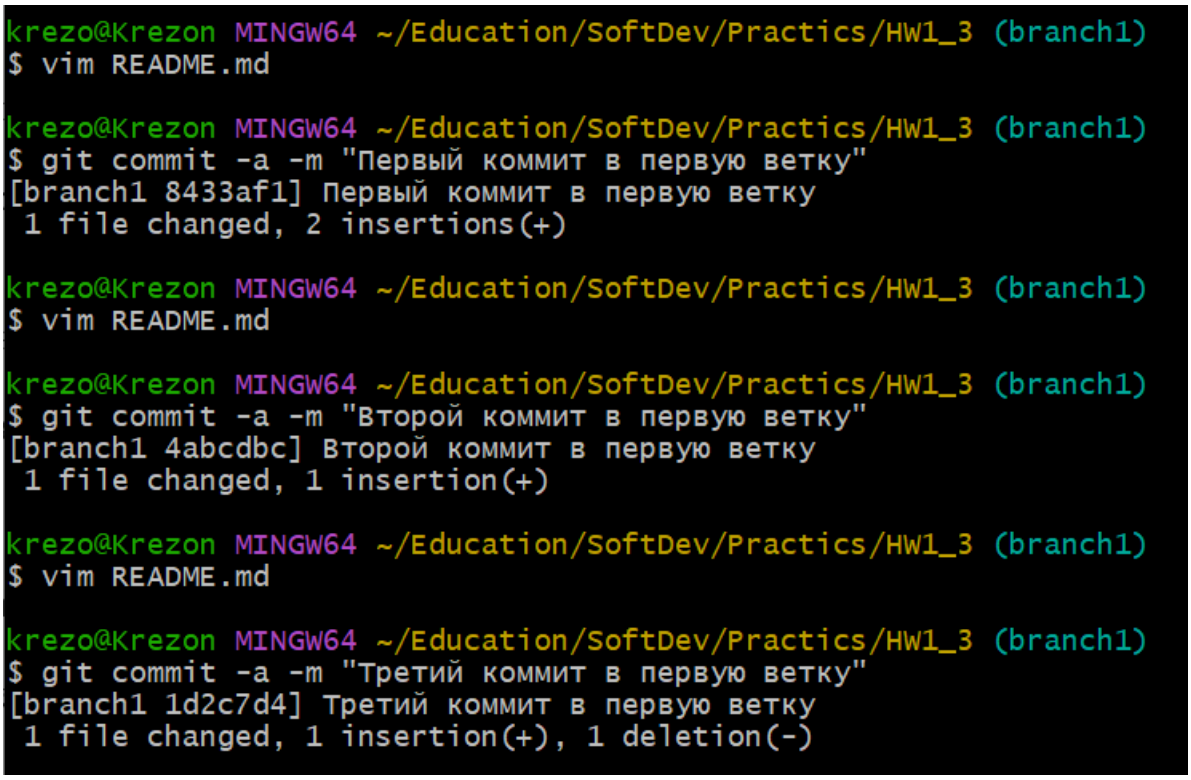

Рисунок 52 – Три коммита в ветку branch1

Сделаем три коммита во вторую ветку (рисунок 53).

```
krezo@Krezon MINGW64 ~/Education/SoftDev/Practics/HW1_3 (branch1)
$ git checkout branch2
Switched to branch 'branch2'

krezo@Krezon MINGW64 ~/Education/SoftDev/Practics/HW1_3 (branch2)
$ vim README.md

krezo@Krezon MINGW64 ~/Education/SoftDev/Practics/HW1_3 (branch2)
$ git commit -a -m "Первый коммит во вторую ветку"
[branch2 d89eab7] Первый коммит во вторую ветку
 1 file changed, 2 insertions(+)

krezo@Krezon MINGW64 ~/Education/SoftDev/Practics/HW1_3 (branch2)
$ vim README.md

krezo@Krezon MINGW64 ~/Education/SoftDev/Practics/HW1_3 (branch2)
$ git commit -a -m "Второй коммит во вторую ветку"
[branch2 8d491d6] Второй коммит во вторую ветку
 1 file changed, 1 insertion(+), 1 deletion(-)

krezo@Krezon MINGW64 ~/Education/SoftDev/Practics/HW1_3 (branch2)
$ vim README.md

krezo@Krezon MINGW64 ~/Education/SoftDev/Practics/HW1_3 (branch2)
$ git commit -a -m "Третий коммит во вторую ветку"
[branch2 63875c6] Третий коммит во вторую ветку
 1 file changed, 2 insertions(+), 1 deletion(-)
```

Рисунок 53 – Три коммита в ветку branch2

Сольем ветку **branch1** в ветку **branch2** (рисунок 54) и разрешим все конфликты (рисунки 55-56).

```
krezo@Krezon MINGW64 ~/Education/SoftDev/Practics/HW1_3 (branch2)
$ git merge branch1
Auto-merging README.md
CONFLICT (content): Merge conflict in README.md
Automatic merge failed; fix conflicts and then commit the result.

krezo@Krezon MINGW64 ~/Education/SoftDev/Practics/HW1_3 (branch2|MERGING)
$ vim README.md

krezo@Krezon MINGW64 ~/Education/SoftDev/Practics/HW1_3 (branch2|MERGING)
$ git commit -a -m "Слияние ветки 1 в ветку 2"
[branch2 6798db6] Слияние ветки 1 в ветку 2

krezo@Krezon MINGW64 ~/Education/SoftDev/Practics/HW1_3 (branch2)
```

Рисунок 54 – Слияние веток branch1 и branch2

```
</sup>

<<<<<<< HEAD
A second change one line back
A first change here as well
=======
A third change instead of the first
And a second change here too
>>>>>>> branch1

![Monty Python](web/image_888.jpeg)
```

Рисунок 55 – Конфликт при слиянии

```
</sup>

A second change one line back
A first change here as well
A third change instead of the first
And a second change here too

![Monty Python](web/image_888.jpeg)
```

Рисунок 56 – Решенный конфликт при слиянии

Выгрузим все изменения в удаленный репозиторий (рисунок 57).

```
krezo@Krezon MINGW64 ~/Education/SoftDev/Practics/HW1_3 (branch2)
$ git push origin --all
Enumerating objects: 23, done.
Counting objects: 100% (23/23), done.
Delta compression using up to 20 threads
Compressing objects: 100% (21/21), done.
Writing objects: 100% (21/21), 2.88 KiB | 984.00 KiB/s, done.
Total 21 (delta 8), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (8/8), completed with 1 local object.
To github.com:Krezerenko/python-cheatsheet.git
 * [new branch]      branch1 -> branch1
 * [new branch]      branch2 -> branch2
```

Рисунок 57 – Выгрузка всех изменений в удаленный репозиторий

Сделаем еще три коммита в ветку **branch1** (рисунок 58).

```
krezo@Krezon MINGW64 ~/Education/SoftDev/Practics/HW1_3 (branch2)
$ git checkout branch1
Switched to branch 'branch1'

krezo@Krezon MINGW64 ~/Education/SoftDev/Practics/HW1_3 (branch1)
$ vim README.md

krezo@Krezon MINGW64 ~/Education/SoftDev/Practics/HW1_3 (branch1)
$ git commit -a -m "Четвертый коммит в первую ветку"
[branch1 a1164c4] Четвертый коммит в первую ветку
 1 file changed, 1 insertion(+)

krezo@Krezon MINGW64 ~/Education/SoftDev/Practics/HW1_3 (branch1)
$ vim README.md

krezo@Krezon MINGW64 ~/Education/SoftDev/Practics/HW1_3 (branch1)
$ git commit -a -m "Пятый коммит в первую ветку"
[branch1 98a759d] Пятый коммит в первую ветку
 1 file changed, 1 insertion(+)

krezo@Krezon MINGW64 ~/Education/SoftDev/Practics/HW1_3 (branch1)
$ vim README.md

krezo@Krezon MINGW64 ~/Education/SoftDev/Practics/HW1_3 (branch1)
$ git commit -a -m "Шестой коммит в первую ветку"
[branch1 b756819] Шестой коммит в первую ветку
 1 file changed, 1 insertion(+), 1 deletion(-)
```

Рисунок 58 – Еще три коммита в ветку branch1

Склонируем репозиторий в другую папку (рисунок 59).

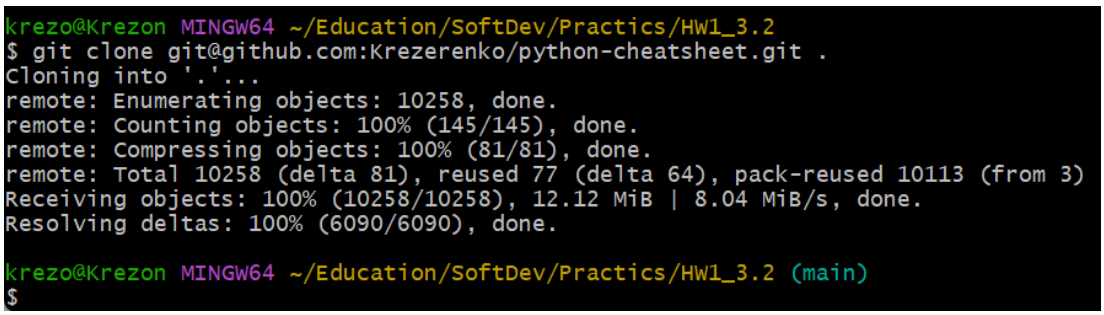

Рисунок 59 – Клонирование репозитория в новую папку

Сделаем три коммита в новом клоне (рисунок 60).

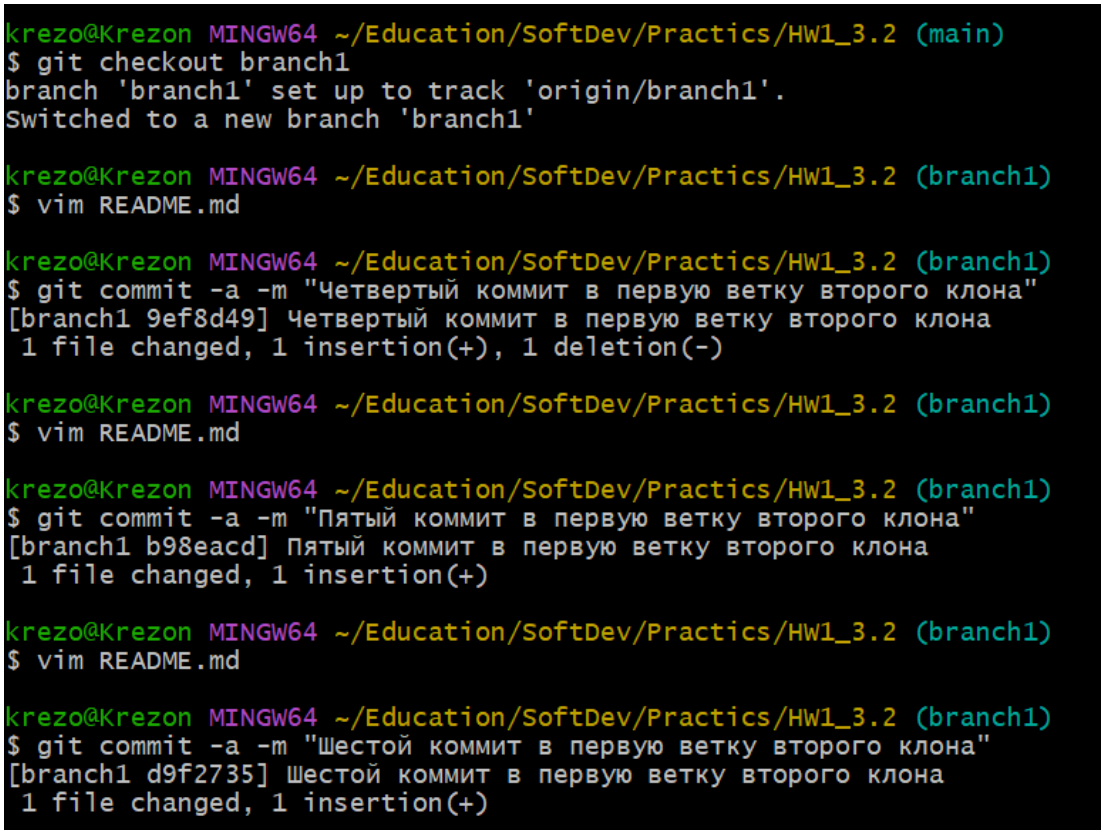

Рисунок 60 – Три коммита в новом клоне

Выгрузим все изменения из нового клона в удаленный репозиторий (рисунок 61).

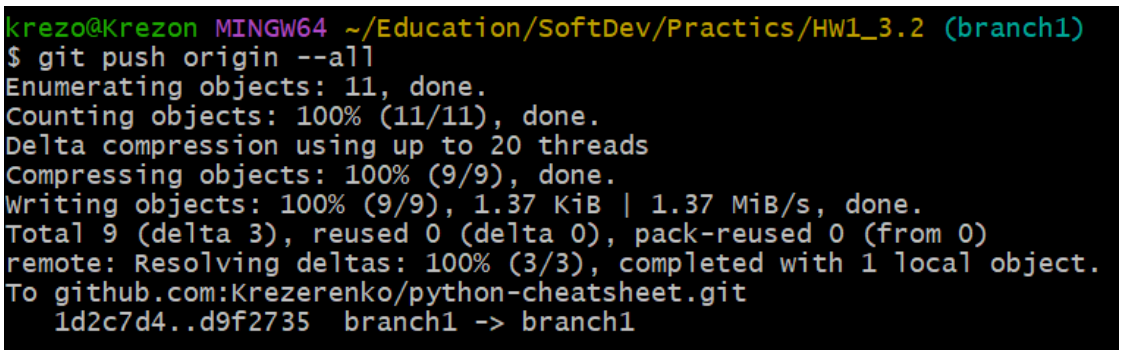

Рисунок 61 – Выгрузка изменений из нового клона

Вернемся в старый клон и выгрузим изменения с опцией **–force** (рисунок 62).

```
krezo@Krezon MINGW64 ~/Education/SoftDev/Practics/HW1_3 (branch1)
$ git push origin --all --force
Enumerating objects: 11, done.
Counting objects: 100% (11/11), done.
Delta compression using up to 20 threads
Compressing objects: 100% (9/9), done.
Writing objects: 100% (9/9), 1.35 KiB | 693.00 KiB/s, done.
Total 9 (delta 3), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (3/3), completed with 1 local object.
To github.com:Krezerenko/python-cheatsheet.git
 + d9f2735...b756819 branch1 -> branch1 (forced update)
```

Рисунок 62 – Выгрузка изменений из старого клона с опцией --force

Загрузим все изменения в новый репозиторий и разрешим конфликты (рисунок 63).

```
krezo@Krezon MINGW64 ~/Education/SoftDev/Practics/HW1_3.2 (branch1)
$ git pull origin
remote: Enumerating objects: 9, done.
remote: Counting objects: 100% (9/9), done.
remote: Compressing objects: 100% (6/6), done.
remote: Total 9 (delta 3), reused 9 (delta 3), pack-reused 0 (from 0)
Unpacking objects: 100% (9/9), 1.33 KiB | 20.00 KiB/s, done.
From github.com:Krezerenko/python-cheatsheet
 + d9f2735...b756819 branch1    -> origin/branch1  (forced update)
Auto-merging README.md
CONFLICT (content): Merge conflict in README.md
Automatic merge failed; fix conflicts and then commit the result.

krezo@Krezon MINGW64 ~/Education/SoftDev/Practics/HW1_3.2 (branch1|MERGING)
$ vim README.md

krezo@Krezon MINGW64 ~/Education/SoftDev/Practics/HW1_3.2 (branch1|MERGING)
$ git commit -a -m "Слил старый репозиторий в новый"
[branch1 3c36a85] Слил старый репозиторий в новый
```

Рисунок 63 – Загрузка изменений в новый клон репозитория и решение конфликтов

3 Ответы на контрольные вопросы

1. Что делает команда git status? — Отображает файлы, которые имеют различия между файлом индекса и текущим коммитом HEAD.
2. Что делает команда git add? — Эта команда обновляет индекс, используя текущее содержимое, найденное в рабочем дереве, чтобы подготовить содержимое для следующего коммита. «Индекс» содержит снапшот содержимого рабочего дерева, и именно этот снимок берется в качестве содержимого следующего коммита. Таким образом, после внесения любых изменений в рабочее дерево и перед запуском команды коммита необходимо использовать команду add для добавления любых новых или измененных файлов в индекс.
3. Что делает команда git log? — Показывает журналы коммитов. Перечисляет коммиты, которые доступны по родительским ссылкам из заданных коммитов, но исключает коммиты, которые доступны из заданных с ^ перед ними. Вывод по умолчанию выдается в обратном хронологическом порядке.
4. Что делает команда git diff? — Показывает изменения между рабочим деревом и индексом или деревом, изменения между индексом и деревом, изменения между двумя деревьями, изменения, полученные в результате слияния, изменения между двумя объектами или изменения между двумя файлами на диске.
5. Что делает команда git show? — Показывает один или несколько объектов (blobs, деревья, теги и коммиты). Для коммитов он показывает сообщение лога и текстовое различие. Он также представляет коммит слияния в специальном формате, как git diff-tree --cc. Для тегов он показывает сообщение тега и ссылочные объекты. Для деревьев он показывает имена (эквивалентно git ls-tree --name-only).

6. Что делает команда git stash? — Используется для записи текущего состояния рабочего каталога и индекса, для возвращения к чистому рабочему каталогу. Команда сохраняет локальные изменения и возвращает рабочий каталог в соответствие с коммитом HEAD. Изменения, сохраненные этой командой, можно просмотреть с помощью git stash list, проверить с помощью git stash show и восстановить (возможно, поверх другого коммита) с помощью git stash apply.
7. Как узнать, кто автор строчки в файле, используя систему Git? git blame — показывает, какой коммит и автор в последний раз изменили каждую строку файла.
8. Как отменить действие команды "git add" на файл? — git-restore — восстанавливает файлы рабочего дерева. Эту команду также можно использовать для восстановления содержимого в индексе с помощью —staged.

ЗАКЛЮЧЕНИЕ

В ходе проделанной работы были изучены основы инструментов управления версиями, запуск и настройка git на своей машине, основные команды для работы с локальными и удалёнными репозиториями git.