

Лабораторна робота № 9

Тема: Робота з історією змін та отримання старих версій.

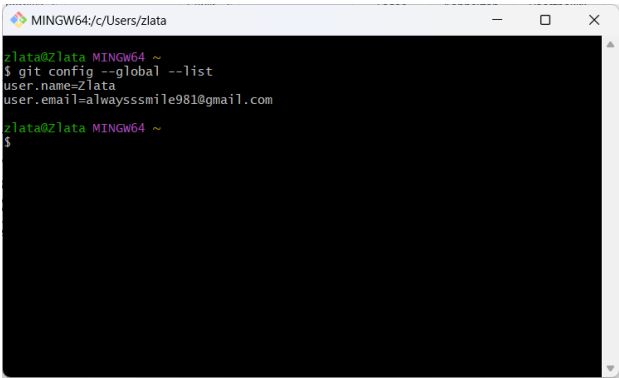
Мета: Отримати навички переміщення по історії змін у СКВ Git.

Обладнання та ПЗ: Персональний комп'ютер

Хід роботи та завдання

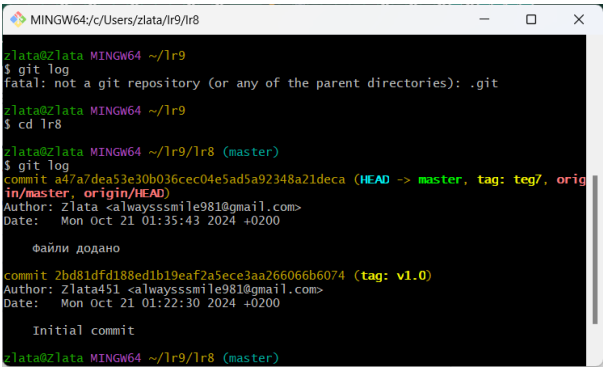
Пункт 1.

Перевірка глобальних налаштувань Git for Windows.



Пункт 3, 4.

Клоновано створений раніше репозиторій як локальну робочу копію та переглянуно історію комітів проекту.



					Лабораторна робота № 9			
Зм	Лист	№ докум	Підпис	Дата	Робота з історією змін та отримання старих версій	Літ.	Лист.	Листів.
Розробив	Ізмestьєва					У	2	4
Перевірів	Левицткий					Група 451		
Оцінка								
Затв								

Пункт 5.

Завантажено одну з попередніх версій проекту через тег teg7.

```
MINGW64~/c/Users/zlata/lr9/lr8
zlata@Zlata MINGW64 ~/lr9/lr8 (master)
$ git checkout teg7
Note: switching to 'teg7'.

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>

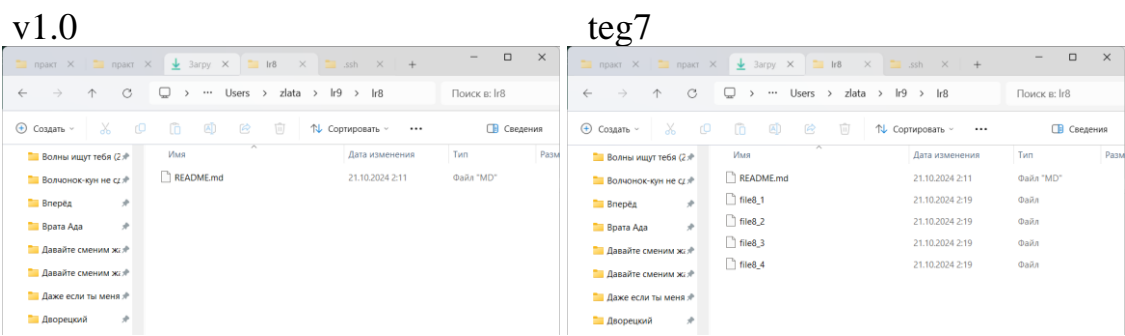
Or undo this operation with:

  git switch -

Turn off this advice by setting config variable advice.detachedHead to false
HEAD is now at a47a7de файли додано
zlata@Zlata MINGW64 ~/lr9/lr8 ((teg7))
$
```

Пункт 6.

Порівняння 2 тегів:



Коли «увімкнено» тег v1.0, то у старій версії відображається початкові зміни репозиторію, тоді як тег teg7 показую останні зміни (нові додані файли).

Пункт 8, 9.

Скасовано останній коміт та переглянуто історію.

```
MINGW64~/c/Users/zlata/lr9/lr8
'git <command> [<revision>...] -- [<file>...]'
zlata@Zlata MINGW64 ~/lr9/lr8 ((v1.0))
$ git reset --soft HEAD^
zlata@Zlata MINGW64 ~/lr9/lr8 ((v1.0))
$ git checkout master
Previous HEAD position was 2bd81df Initial commit
Switched to branch 'master'
Your branch is up to date with 'origin/master'.

zlata@Zlata MINGW64 ~/lr9/lr8 (master)
$ git reset --soft HEAD^
zlata@Zlata MINGW64 ~/lr9/lr8 (master)
$ git log
commit 2bd81dfd188ed1b19eaf2a5ece3aa266066b6074 (HEAD -> master, tag: v1.0)
Author: Zlata451 <alwayssmile981@gmail.com>
Date: Mon Oct 21 01:22:30 2024 +0200

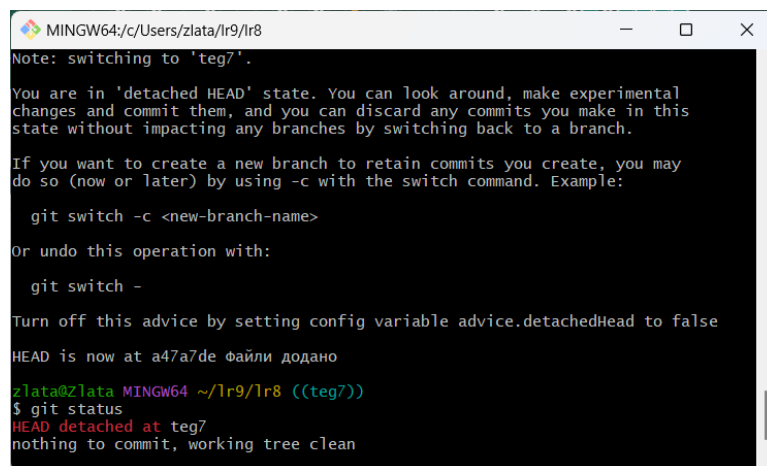
    Initial commit

zlata@Zlata MINGW64 ~/lr9/lr8 (master)
$
```

					Лабораторна робота № 9	Арк.
						4
Зм.	Арк.	№ докум.	Підпис	Дата		

Пункт 11.

Завантажено репозиторій у стані обраного попереднього коміту (teg7).



```
MINGW64: c:/Users/zlata/lr9/lr8
Note: switching to 'teg7'.

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 a47a7de Файли додано
zlata@Zlata MINGW64 ~/lr9/lr8 ((teg7))
$ git status
HEAD detached at teg7
nothing to commit, working tree clean
```

Контрольні питання

1. Скільки існує основних способів вибору ревізій?

Існує два основних способи вибору ревізій в Git: за допомогою хешу коміту та за допомогою тегів. Також можна використовувати гілки для навігації між ревізіями.

2. Скільки символів зазвичай достатньо для збереження унікальності значень в проєкті?

Зазвичай 7-10 символів SHA-1 хешу коміту є достатніми для збереження унікальності значень в проєкті. Однак, для більшої точності, зазвичай рекомендується використовувати повний хеш.

3. За допомогою якої команди можливо продивитися історію комітів?

Команда `git log` дозволяє переглядати історію комітів.

4. Перелічіть відомі вам опції команди `git log`.

Декілька основних опцій команди **git log**:

1. `--oneline` – відображає коміти в одному рядку.
2. `--graph` – показує графічне зображення гілок та комітів.
3. `--decorate` – показує теги та гілки разом з комітами.
4. `--author="ім'я"` – фільтрує коміти за автором.
5. `--since` та `--until` – дозволяють фільтрувати коміти за датою.

					Лабораторна робота № 9	Арк.
						4
Зм.	Арк.	№ докум.	Підпис	Дата		

5. Опишіть опції команди `git log`.

--oneline: Відображає короткий формат комітів, зменшуючи кількість інформації до одного рядка для кожного коміту.

--graph: Відображає візуальний граф гілок, що полегшує розуміння структури комітів.

--decorate: Додає до виходу інформацію про теги та гілки, що вказують на певні коміти.

--author="ім'я": Дозволяє фільтрувати результати команди за конкретним автором, що спрощує пошук комітів, зроблених певною особою.

--since і **--until:** Використовуються для обмеження виводу комітів в залежності від дати, що корисно для аналізу змін у певний період часу.

Висновок: У ході лабораторної роботи я отримала навички переміщення по історії змін у СКВ Git

					<i>Лабораторна робота № 9</i>	Арк.
						4
Зм.	Арк.	№ докум.	Підпис	Дата		