Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра інформаційних систем та технологій

Лабораторна робота № 1

з дисципліни «Технології розроблення програмного забезпечення»

Тема: «Основи роботи з Git»

Виконала:
студентка групи ІА-33
Котик Іванна.
Дата здачі 06.09.2025

Перевірив:
асистент кафедри ІСТ
Мягкий Михайло Юрійович

Київ 2025

Тема: Основи роботи з системою контролю версій Git: ініціалізація репозиторію, створення гілок, виконання комітів і злиття.

Мета: Навчитися працювати з базовими командами Git для керування версіями проєкту.

Хід роботи:

1. Створюємо директорію за допомогою команди **mkdir** з назвою “new”.

```
C:\Users\KOTIK>mkdir new
```

2. Перейдемо в директорію “new”, використовуючи команду **cd.** Усі наступні дії виконуються в цьому каталозі.

```
C:\Users\KOTIK>cd new
```

3. Створимо новий файл під назвою “f1.txt” і запишемо у нього дані (у нашому випадку число 1), використовуючи команду **echo "1" > f1.txt.**

   *Також можна скористатися командою **touch f1.txt**, якщо використовувати Linux.*

```
C:\Users\KOTIK\new>echo "1" > f1.txt
```

4. Ініціалізуємо Git-репозиторію в поточній папці.

```
C:\Users\KOTIK\new>git init
Initialized empty Git repository in C:/Users/KOTIK/new/.git/
```

5. Перевіримо стан репозиторію за допомогою команди **git status**.

```
C:\Users\KOTIK\new>git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        f1.txt

nothing added to commit but untracked files present (use "git add" to track)
```

Git показує, що є не відслідковуваний файл “f1.txt”.

6. За допомогою команди **git add .** додаємо усі файли у область підготовки до коміту. Та, використовуючи команду **git commit -m,** ствоюємо перший коміт з назвою “init”. У коміт потрапляє файл “f1.txt”.

```
C:\Users\KOTIK\new>git add .

C:\Users\KOTIK\new>git commit -m "init"
[master (root-commit) 86a5a48] init
 1 file changed, 1 insertion(+)
 create mode 100644 f1.txt
```

7. Створюємо нові гілки різними командами:

    а) **git branch** b2 – створить нову гілку з назвою “b2”;

    б) **git checkout -b** b1 – створисть і одночасно перейде у гілку “b1”.

```
C:\Users\KOTIK\new>git branch b2

C:\Users\KOTIK\new>git checkout -b b1
Switched to a new branch 'b1'
```

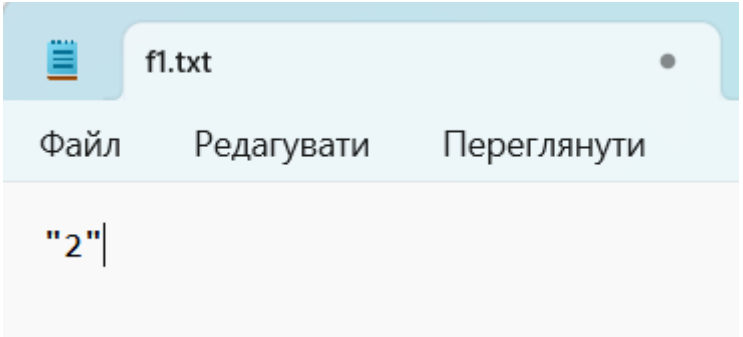
8. Використовуємо команду **git log** , щоб переглянути історію змін комітів.

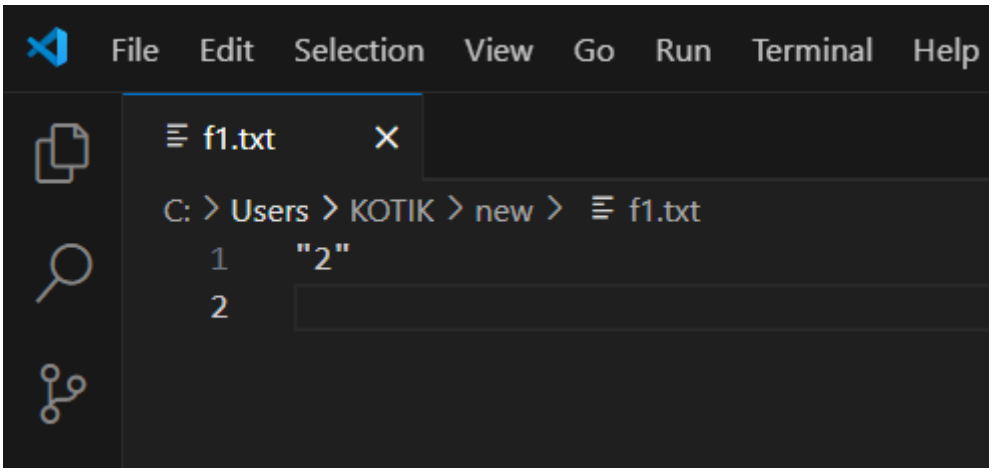
```
C:\Users\KOTIK\new>git log
commit 86a5a4830f6201104bbc195b41bbe70322a13381 (HEAD -> b1, master, b2)
Author: ivannakotyk <ivannakotik3@gmail.com>
Date:   Sat Sep 6 13:49:31 2025 +0300

    init
```

Ми переконалися, що коміт “init” є в гілках master, b1 і b2, тобто усі гілки були успішно створені. Поточна гілка – b1.

9. Змінемо вміст файла “f1.txt” (щоб відкрити файли одразу можемо скористатися командами **notepad** f1.txt (редактор – блокнот) або **code** f1.txt (редактор – Visual Studio Code)).





10. Для перевірки внесених змін у файл скористаємося командою **git status.**

```
C:\Users\KOTIK\new>git status
On branch b1
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:   f1.txt

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

Ми переконалися, що зміни були внесені, і не додані до коміту.

11. Закомітимо наші зміни, використовуючи команду **git commit -am** “modify”. Символ **-a** автоматично додає до коміту всі змінені файли, які вже відслідковуються Git.

```
C:\Users\KOTIK\new>git commit -am "modify"
[b1 b98f96f] modify
 1 file changed, 1 insertion(+), 1 deletion(-)
```

12. Перейдемо на гілку b2 за допомогою команди **git checkout**.

```
C:\Users\KOTIK\new>git checkout b2
Switched to branch 'b2'
```

13. Зливаєм гілку b1 в поточну гілку за допомогою команди **git merge** b1.

```
C:\Users\KOTIK\new>git merge b1
Updating 86a5a48..b98f96f
Fast-forward
 f1.txt | 2 +-
 1 file changed, 1 insertion(+), 1 deletion(-)
```

Git повідомив про **fast-forward**. Це означає, що гілка b2 відставала від гілки b1 і не мала власних унікальних комітів. Тому Git просто пересунув вказівник b2 на останній коміт з гілки b1.

14. Використовуючи команду **git log --graph**, переглянемо історію комітів у вигляді графа.

```
C:\Users\KOTIK\new>git log --graph
* commit b98f96fcca71e636a4a258c54e42585a8780c456 (HEAD -> b2, b1)
| Author: ivannakotyk <ivannakotik3@gmail.com>
| Date:   Sat Sep 6 13:57:14 2025 +0300
|
|     modify
|
* commit 86a5a4830f6201104bbc195b41bbe70322a13381 (master)
  Author: ivannakotyk <ivannakotik3@gmail.com>
  Date:   Sat Sep 6 13:49:31 2025 +0300

      init
```

Таким чином, гілки b1 і b2 синхронізовані між собою (обидві вказують на коміт “modify”), а гілка master поки що відстає і містить лише початковий коміт “init”.

Висновки:

В ході виконання лабораторної роботи ми ознайомилися з основами роботи з Git і навчилася користуватися її базовими командами. Було здійснено ініціалізацію репозиторію, створення та перемикання між гілками, внесення змін у файли та комітування цих змін. Також ми відпрацювали процес злиття гілок та перегляду історії комітів у вигляді графа, що дозволило наочно побачити взаємозв’язок між гілками та комітами.