Міністерство освіти і науки України
Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”
Факультет інформатики та обчислювальної техніки
Кафедра автоматики та управління в технічних системах

Лабораторна робота №1
Технологія розробки програмного забезпечення
«Git»

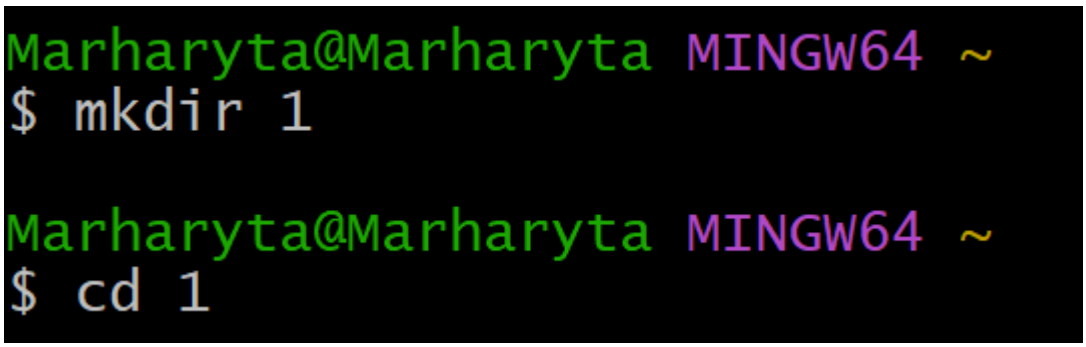
Виконала
студентка групи ІА–32:
Ткачук М. С.

Перевірив:
Мягкий М. Ю.

Київ 2025

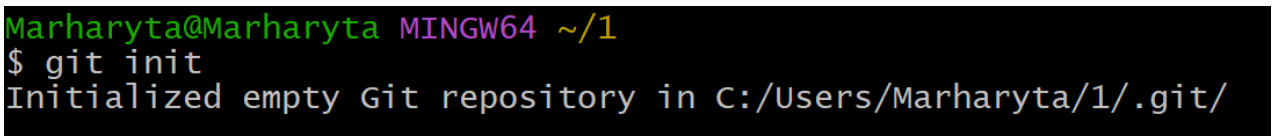
Зміст

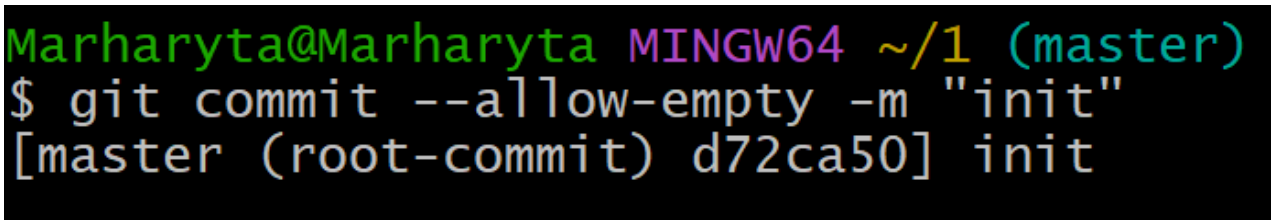
- mkdir 1
- cd 1
- git init
- git commit --allow-empty -m "init"
- git branch b1
- git checkout -b b2
- git checkout -
- git switch -c b3
- git checkout -
- touch f1.txt f2.txt f3.txt
- git add .
- git status
- git commit -m "f1" f1.txt
- git checkout master
- git commit -m "commit f2" f2.txt
- git commit -m "commit f3" f3.txt
- git merge b1
- echo "a" > f1.txt
- git add .
- git commit -m "added f1 with some data"
- git merge b2
- git log --graph



mkdir 1 - створити нову директорію з назвою *1* у поточній робочій директорії

cd 1 - перейти в створену директорію *1*





git init - ініціалізувати новий Git-репозиторій у папці 1

git commit --allow-empty -m "init" - створити початковий (порожній) коміт "init"

```
Marharyta@Marharyta MINGW64 ~/1 (master)
$ git branch b1

Marharyta@Marharyta MINGW64 ~/1 (master)
$ git checkout -b b2
Switched to a new branch 'b2'

Marharyta@Marharyta MINGW64 ~/1 (b2)
$ git checkout -
Switched to branch 'master'

Marharyta@Marharyta MINGW64 ~/1 (master)
$ git switch -c b3
Switched to a new branch 'b3'

Marharyta@Marharyta MINGW64 ~/1 (b3)
$ git checkout -
Switched to branch 'master'
```

git branch b1 - створити нову гілку b1

git checkout -b b2 - створити гілку b2 і відразу переключитися на неї

git checkout - – повертаємося назад в master branch, щоб від нього створити ще гілку

git switch -c b3 - створити гілку b3 і відразу перейти на неї

git switch - альтернатива git checkout для роботи з гілками

git checkout - - знову повернулися в master

```
Marharyta@Marharyta MINGW64 ~/1 (master)
$ touch f1.txt f2.txt f3.txt
```

touch f1.txt f2.txt f3.txt - створити три порожні файли f1.txt, f2.txt, f3.txt

```
Marharyta@Marharyta MINGW64 ~/1 (master)
$ git add .

Marharyta@Marharyta MINGW64 ~/1 (master)
$

Marharyta@Marharyta MINGW64 ~/1 (master)
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   f1.txt
        new file:   f2.txt
        new file:   f3.txt
```

git add . - почати відслідковувати файли
git status - показати поточний стан репозиторію. Показує таку інформацію на якій гілці ти зараз знаходишся,

чи є закомічені або не закомічені зміни,

які файли відстежуються, які ще ні,

які файли готові до коміту (staged)

```
Marharyta@Marharyta MINGW64 ~/1 (master)
$ git checkout b1
A       f1.txt
A       f2.txt
A       f3.txt
Switched to branch 'b1'

Marharyta@Marharyta MINGW64 ~/1 (b1)
$ git commit -m "f1" f1.txt
[b1 d8ebac7] f1
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 f1.txt
```

Переходимо на гілку b1 і комітимо тільки f1.txt з усіх файлів, що відслідковуємо

git commit -m "f1" f1.txt - створити коміт, додаючи до репозиторію файл f1.txt

Результат: файл f1.txt закомічений у гілці b1

```
Marharyta@Marharyta MINGW64 ~/1 (b1)
$ git log
commit d8ebac77f141b0915a59d91166cd8da4326e5cc7 (HEAD -> b1)
Author: Margarita Tkachuk <tk.m.andarinchik@example.com>
Date:   Sat Sep 6 11:11:02 2025 +0300

    f1

commit d72ca50e7931b817407ff6c3bd6f5baaac30ef7f (master, b3, b2)
Author: Margarita Tkachuk <tk.m.andarinchik@example.com>
Date:   Sat Sep 6 11:07:31 2025 +0300

    init
```

```
Marharyta@Marharyta MINGW64 ~/1 (b1)
$ git checkout master
A       f2.txt
A       f3.txt
Switched to branch 'master'
```

Повертаємося назад в master

```
Marharyta@Marharyta MINGW64 ~/1 (master)
$ git checkout b2
A       f2.txt
A       f3.txt
Switched to branch 'b2'

Marharyta@Marharyta MINGW64 ~/1 (b2)
$ git commit -m "commit f2" f2.txt
[b2 2183f59] commit f2
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 f2.txt
```

З master переходимо на b2

git commit -m "commit f2" f2.txt -, створити коміт на гілці b2, додаючи файл f2.txt

```
Marharyta@Marharyta MINGW64 ~/1 (b2)
$ git checkout master
A       f3.txt
Switched to branch 'master'
g
Marharyta@Marharyta MINGW64 ~/1 (master)
$ git checkout b3
A       f3.txt
Switched to branch 'b3'
g
Marharyta@Marharyta MINGW64 ~/1 (b3)
$ git commit -m "commit f3" f3.txt
[b3 f75a9fd] commit f3
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 f3.txt
```


```
Marharyta@Marharyta MINGW64 ~/1 (b3)
$ git commit -m "commit f3" f3.txt
[b3 f75a9fd] commit f3
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 f3.txt
```


```
Marharyta@Marharyta MINGW64 ~/1 (b3)
$ git checkout master
Switched to branch 'master'
```

Повертаємося в master і з нього переходимо на b3, маємо останній файл, що не є закоміченим

git commit -m "commit f3" f3.txt - створити коміт на гілці b3, додаючи файл f3.txt

і повертаємося назад в master

```
Marharyta@Marharyta MINGW64 ~/1 (master)
$ git checkout b2
Switched to branch 'b2'
```

```
Marharyta@Marharyta MINGW64 ~/1 (b2)
$ touch f1.txt
```

З master переходимо в b2 і створюємо файл f1.txt, який вже створений в гілці b1

(тут історія з терміналу не зберіглася, тому текстом пишу)
комітимо файл f1.txt в гілці b2
і виконуємо

git merge b1 - злити зміни з гілки b1 у гілку b2

Результат: з’явився merge-коміт, що об’єднав файли f1.txt і f2.txt

```
Merge made by the 'ort' strategy.

Marharyta@Marharyta MINGW64 ~/1 (b2)
$ git log
commit 382519c8a64f7cf1d4cfed4284bbba16dbce35b3 (HEAD -> b2)
Merge: 8682646 d8ebac7
Author: Margarita Tkachuk <tk.m.andarinchik@example.com>
Date:   Sat Sep 6 11:16:34 2025 +0300

    Merge branch 'b1' into b2
```

```
Marharyta@Marharyta MINGW64 ~/1 (b2)
$ git checkout master
Switched to branch 'master'

Marharyta@Marharyta MINGW64 ~/1 (master)
$ git checkout b3
Switched to branch 'b3'
```

Повертаємося з b2 в master і з нього переходимо в b3

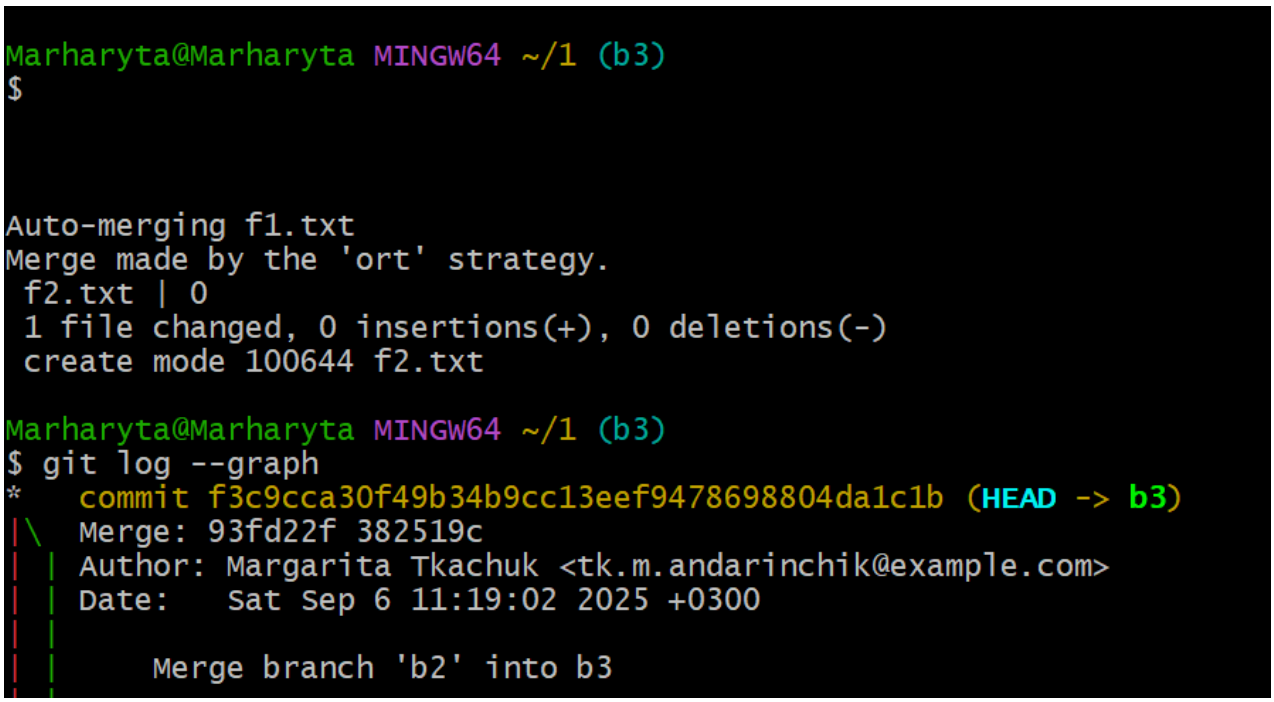
```
Marharyta@Marharyta MINGW64 ~/1 (b3)
$ echo "a" > f1.txt

Marharyta@Marharyta MINGW64 ~/1 (b3)
$ cat f1.txt
a

Marharyta@Marharyta MINGW64 ~/1 (b3)
$

Marharyta@Marharyta MINGW64 ~/1 (b3)
$ git add .
warning: in the working copy of 'f1.txt', LF will be replaced by CRLF the next time Git touches it

Marharyta@Marharyta MINGW64 ~/1 (b3)
$ git commit -m "added f1 with some data"
[b3 93fd22f] added f1 with some data
 1 file changed, 1 insertion(+)
 create mode 100644 f1.txt
```

echo "a" > f1.txt

git add .

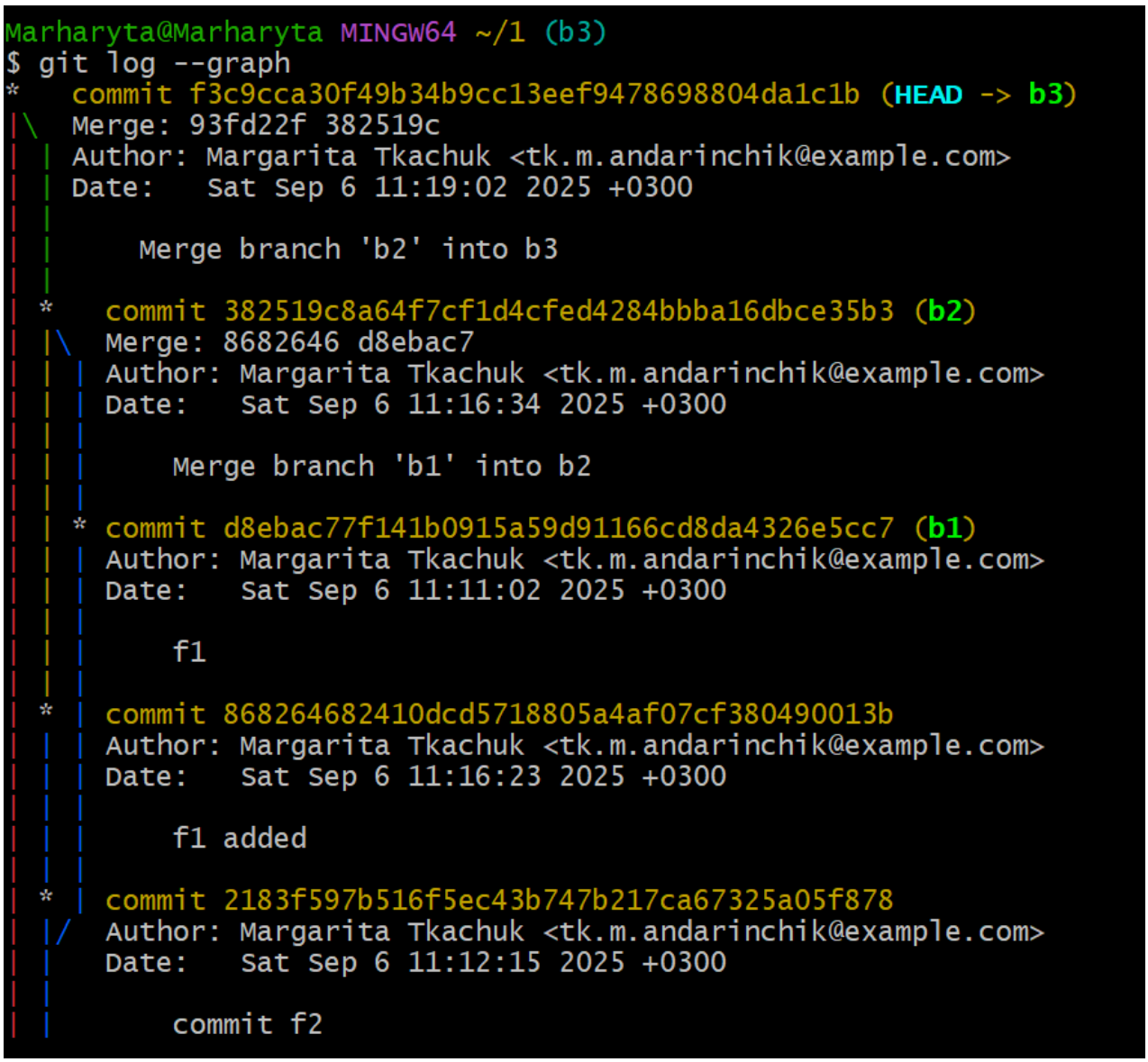
git commit -m "added f1 with some data"

задача команд – створити файл f1.txt з змістом “a”, і закомітити цей файл



git merge b2 - злити зміни з гілки b2 у гілку b3

В гілці b2 вже був f1.txt, але він був пустий, тож той файл замінився на f1.txt з гілки b3, бо він мав зміст



git log –graph - показати графічну структуру комітів у репозиторії.

Результат: видно дерево злиттів між гілками *b1, b2, b3*.

Висновок: У ході виконання лабораторної роботи було опрацьовано основні команди системи контролю версій Git. Я створила новий репозиторій, додала в нього файли та організувала роботу з кількома гілками (b1, b2, b3). На практиці було показано, як створювати коміти для окремих файлів, об’єднувати гілки за допомогою команди git merge, а також як переглядати історію змін за допомогою git log --graph.