Міністерство освіти і науки України
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: «Системи контроля версій. Розподілена система контролю версій «Git»»

Виконав
студент групи ІА–33
Марченко Вадим Олександрович

Київ 2025

Тема: Системи контроля версій. Розподілена система контролю версій «Git»
Мета: Навчитися виконувати основні операції в роботі з децентралізованими системами контролю версій на прикладі роботи з сучасною системою Git.

Короткі теоретичні відомості

Система управління версіями – програмне забезпечення яке призначено допомогти команді розробників керувати змінами в вихідному коді під час роботи. Система керування версіями дозволяє додавати зміни в файлах в репозиторій і таким чином після кожної фіксації змін мани нову ревізію файлів. Це дозволяє повертатися до попередніх версій коду для аналізу внесених змін або пошуку, які зміни призвели до появи помилки. Таким чином можна знайти хто, коли і які зміни зробив в коді, а також чому ці зміни були зроблені.

Умовно, розвиток систем контролю версій можна розбити на наступні етапи: ранній етап, етап централізованих систем, етап децентралізації та етап хмарних платформ.

Ранній етап: На цьому етапі основна увага приділялася роботі з окремими файлами у локальному середовищі. Найпершою системою контролю версій була система «скопіювати і вставити», коли більшість проєктів просто копіювалася з місця на місце зі зміною назва (проєкт_1; проєкт_новий; проєкт_найновіший і т.д.), як правило у вигляді zip архіву або подібних (arj, tar).

Етап централізованих систем: У цей період розробники почали переходити до централізованих систем, що дозволяли працювати кільком користувачам одночасно через сервер. Одина із перших найпопулярніших систем (і досі використовувана) система контролю версій – CVS. Цю епоху можна охарактеризувати досить сформованим уявленням про системи контролю версій, їх можливості, появою центральних репозиторіїв (та синхронізації дій команди).

Етап децентралізації: Децентралізовані системи усунули залежність від центрального сервера та дозволили кожному розробнику мати повну копію репозиторію. У 1992 році з'явився один з основних представників світу систем розподіленого контролю версій. ClearCase був однозначно попереду свого часу і для багатьох він досі є однією з найпотужніших систем контролю версій будьколи створених.

Лінус Торвальдс, т.зв. Батько Лінукса, розробив і впровадив першу версію Гіт для надання можливості розробникам ядра Лінукс проводити контроль версій не тільки в BitKeeper. Гіт є системою розподіленого контролю версій, коли кожен розробник має власний репозиторій, куди він вносить зміни. Далі система гіт синхронізує репозиторії із центральним

репозиторієм. Це дозволяє проводити роботу незалежно від центрального репозиторію (на відміну від SVN, коли версіонування передбачало наявність зв'язку з центральним сервером), перекладає складності ведення гілок та склеювання змін більше на плечі системи, ніж розробників та ін.

Етап хмарних платформ: У сучасну епоху акцент робиться на інтеграції систем контролю версій із хмарними платформами та автоматизації розробки. І в більшості випадків такою системою контролю версій є Git. Можна виділити такі ключові хмарні платформи на основі Git: GitHub, GitLab, Bitbucket. Вони підтримують CI/CD, спільну роботу та інтеграції, інструменти для DevOps, аналітики та автоматичного тестування.

Робота з Git може виконуватися з командного рядка, а також за допомогою візуальних оболонок. Командний рядок використовується програмістами, як можливість виконання всіх доступних команд, а також можливості складання складних макросів. Візуальні оболонки як правило дають більш наглядне представлення репозиторію у вигляді дерева, та більш зручний спосіб роботи з репозиторієм, але, дуже часто доступний не весь набір команд Git, а лише саме ті, що найчастіше використовуються.

Відповідно, є ряд основних команд для роботи:

1. Клонувати репозиторій (git clone) – отримати копію репозиторію на локальну машину для подальшої роботи з ним;
2. Синхронізація репозиторіїв (git fetch або git pull) – отримання змін із віддаленого (вихідного, центрального, або будь-якого іншого такого ж) репозиторію;
3. Фіксація змін в репозиторій (git commit) – фіксація виконаних змін в програмному коді в локальний репозиторій розробника;
4. Синхронізація репозиторіїв (git push) – переслати зміни – push – передача власних змін до віддаленого репозиторію – Записати зміни – commit – створення нової версії;
5. Оновитись до версії – update – оновитись до певної версії, що є у репозиторії.
6. Об’єднання гілок (git merge) – об’єднання вказаною гілки в поточну (часто ще називається «злиттям»).

Хід роботи

Створити новий Git репозиторій. Створити три гілки на базі гілки master. До кожної гілки закомітити окремі три файли відповідно: file1.txt, file2.txt, file3.txt. Додати до вітки br3 файли file1.txt, file2.txt. Виконати rebase гілки br3 на br1, розв’язати конфлікти та продовжити rebase. Виконати merge гілки br2 у br3, також розв’язати конфлікти й продовжити merge.

1. Створення Git репозиторію та трьох гілок:

```
Microsoft Windows [Version 10.0.26100.6584]
(c) Корпорація Майкрософт. Усі права захищені.

C:\Users\User>mkdir lab1

C:\Users\User>cd lab1

C:\Users\User\lab1>git init
Initialized empty Git repository in C:/Users/User/lab1/.git/

C:\Users\User\lab1>git status
On branch master

No commits yet

nothing to commit (create/copy files and use "git add" to track)

C:\Users\User\lab1>git commit --allow-empty
[master (root-commit) 4cb2dfb] initial

C:\Users\User\lab1>git branch br1

C:\Users\User\lab1>git checkout -b br2
Switched to a new branch 'br2'

C:\Users\User\lab1>git switch master
Switched to branch 'master'

C:\Users\User\lab1>git switch -c br3
Switched to a new branch 'br3'

C:\Users\User\lab1>git branch
  br1
  br2
* br3
  master

C:\Users\User\lab1>git switch master
Switched to branch 'master'
```

2. Додавання трьох текстових файлів до відповідних гілок:

```
C:\Users\User\lab1>git switch master
Switched to branch 'master'

C:\Users\User\lab1>echo "1" > file1.txt
```

```
C:\Users\User\lab1>echo "1" > file1.txt

C:\Users\User\lab1>echo "2" > file2.txt

C:\Users\User\lab1>echo "3" > file3.txt

C:\Users\User\lab1>git add .

C:\Users\User\lab1>git switch br1
A       file1.txt
A       file2.txt
A       file3.txt
Switched to branch 'br1'

C:\Users\User\lab1>git commit -m "first" file1.txt
[br1 a6cc3c8] first
 1 file changed, 1 insertion(+)
 create mode 100644 file1.txt

C:\Users\User\lab1>git status
On branch br1
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   file2.txt
        new file:   file3.txt


C:\Users\User\lab1>git switch br2
A       file2.txt
A       file3.txt
Switched to branch 'br2'

C:\Users\User\lab1>git commit -m "2" file2.txt
[br2 289fd69] 2
 1 file changed, 1 insertion(+)
 create mode 100644 file2.txt

C:\Users\User\lab1>git switch br3
A       file3.txt
Switched to branch 'br3'
```

```
C:\Users\User\lab1>git commit -m "3" file3.txt
[br3 2455054] 3
 1 file changed, 1 insertion(+)
 create mode 100644 file3.txt
```

3: Додати до вітки br3 файли file1.txt, file2.txt:

```
C:\Users\User\lab1>echo "B" > file1.txt

C:\Users\User\lab1>git add .

C:\Users\User\lab1>git commit -m "gg"
[br3 5b56dd3] gg
 1 file changed, 1 insertion(+)
 create mode 100644 file1.txt
```

```
C:\Users\User\lab1>echo "A" > file2.txt

C:\Users\User\lab1>git add .

C:\Users\User\lab1>git commit -m "A"
```

4. Спроба виконати rebase, merge

```
C:\Users\User\lab1>git rebase br1
Auto-merging file1.txt
CONFLICT (add/add): Merge conflict in file1.txt
error: could not apply 5b56dd3... gg
hint: Resolve all conflicts manually, mark them as resolved with
hint: "git add/rm <conflicted_files>", then run "git rebase --continue".
hint: You can instead skip this commit: run "git rebase --skip".
hint: To abort and get back to the state before "git rebase", run "git rebase --abort".
hint: Disable this message with "git config set advice.mergeConflict false"
Could not apply 5b56dd3... gg

C:\Users\User\lab1>code .

C:\Users\User\lab1>git status
interactive rebase in progress; onto a6cc3c8
Last commands done (2 commands done):
   pick 2455054 3
   pick 5b56dd3 gg
Next command to do (1 remaining command):
   pick 900c81c A
  (use "git rebase --edit-todo" to view and edit)
You are currently rebasing branch 'br3' on 'a6cc3c8'.
  (fix conflicts and then run "git rebase --continue")
  (use "git rebase --skip" to skip this patch)
  (use "git rebase --abort" to check out the original branch)

Unmerged paths:
  (use "git restore --staged <file>..." to unstage)
  (use "git add <file>..." to mark resolution)
        both added:      file1.txt

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

```
C:\Users\User\lab1>git merge br2
Auto-merging file2.txt
CONFLICT (add/add): Merge conflict in file2.txt
Automatic merge failed; fix conflicts and then commit the result.
```

5. Виправлення конфліктів:
Виконавши команду code . запускаємо відповідний файл і залишаємо будь-яке текстове повідомлення. Наприклад, у file1.txt - 1A, file2.txt - 2B.

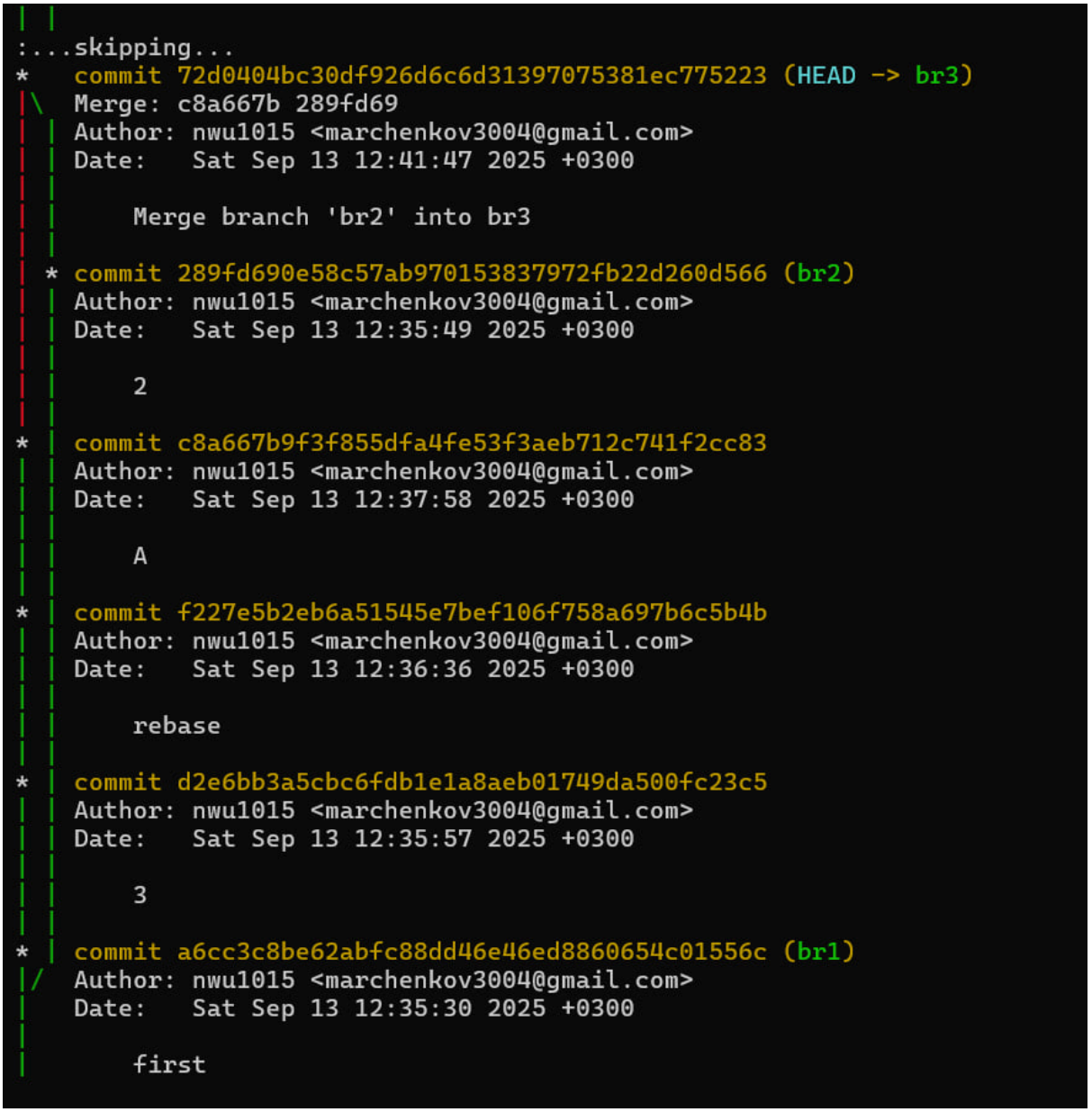
6. Продовжуємо merge і rebase

```
C:\Users\User\lab1>git rebase --continue
[detached HEAD f227e5b] rebase
 1 file changed, 1 insertion(+), 1 deletion(-)
Successfully rebased and updated refs/heads/br3.
```

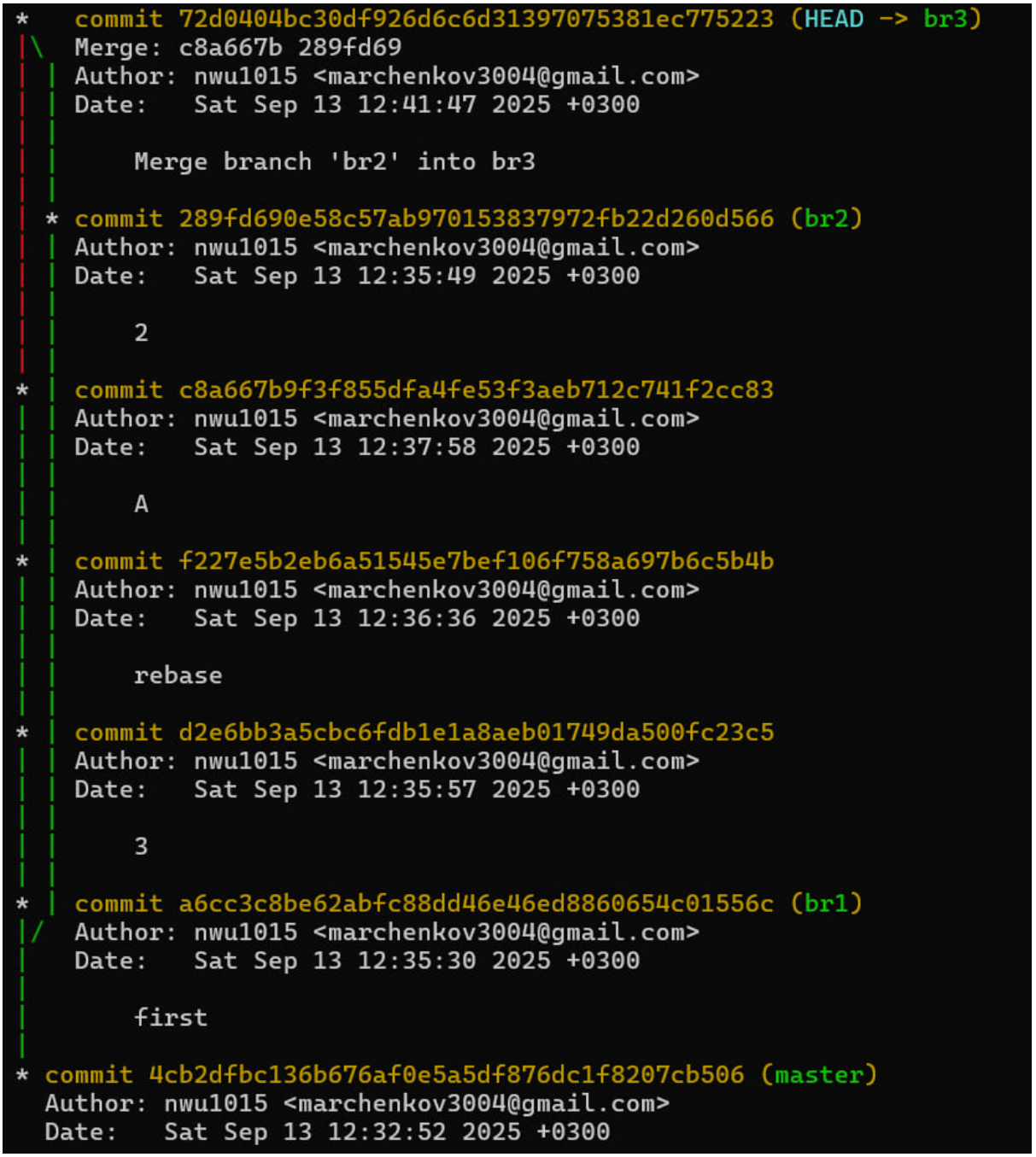
```
C:\Users\User\lab1>git merge --continue
[br3 72d0404] Merge branch 'br2' into br3
```

7. Перевірка логів:

```
C:\Users\User\lab1>git log --all --graph
*   commit 72d0404bc30df926d6c6d31397075381ec775223 (HEAD -> br3)
|\  Merge: c8a667b 289fd69
| | Author: nwu1015 <marchenkov3004@gmail.com>
| | Date:   Sat Sep 13 12:41:47 2025 +0300
| |
| |     Merge branch 'br2' into br3
| |
| * commit 289fd690e58c57ab970153837972fb22d260d566 (br2)
| | Author: nwu1015 <marchenkov3004@gmail.com>
| | Date:   Sat Sep 13 12:35:49 2025 +0300
| |
| |     2
| |
* | commit c8a667b9f3f855dfa4fe53f3aeb712c741f2cc83
| | Author: nwu1015 <marchenkov3004@gmail.com>
| | Date:   Sat Sep 13 12:37:58 2025 +0300
| |
| |     A
| |
* | commit f227e5b2eb6a51545e7bef106f758a697b6c5b4b
| | Author: nwu1015 <marchenkov3004@gmail.com>
| | Date:   Sat Sep 13 12:36:36 2025 +0300
| |
| |     rebase
| |
* | commit d2e6bb3a5cbc6fdb1e1a8aeb01749da500fc23c5
| | Author: nwu1015 <marchenkov3004@gmail.com>
| | Date:   Sat Sep 13 12:35:57 2025 +0300
| |
:...skipping...
*   commit 72d0404bc30df926d6c6d31397075381ec775223 (HEAD -> br3)
|\  Merge: c8a667b 289fd69
| | Author: nwu1015 <marchenkov3004@gmail.com>
| | Date:   Sat Sep 13 12:41:47 2025 +0300
| |
| |     Merge branch 'br2' into br3
```


| |
:...skipping...
* commit 72d0404bc30df926d6c6d31397075381ec775223 (HEAD -> br3)
|\ Merge: c8a667b 289fd69
| | Author: nwu1015 <marchenkov3004@gmail.com>
| | Date: Sat Sep 13 12:41:47 2025 +0300
| |
| | Merge branch 'br2' into br3
| |
| * commit 289fd690e58c57ab9701538379 72fb22d260d566 (br2)
| | Author: nwu1015 <marchenkov3004@gmail.com>
| | Date: Sat Sep 13 12:35:49 2025 +0300
| |
| | 2
| |
* | commit c8a667b9f3f855dfa4fe53f3aeb712c741f2cc83
| | Author: nwu1015 <marchenkov3004@gmail.com>
| | Date: Sat Sep 13 12:37:58 2025 +0300
| |
| | A
| |
* | commit f227e5b2eb6a51545e7bef106f758a697b6c5b4b
| | Author: nwu1015 <marchenkov3004@gmail.com>
| | Date: Sat Sep 13 12:36:36 2025 +0300
| |
| | rebase
| |
* | commit d2e6bb3a5cbc6fdb1e1a8aeb01749da500fc23c5
| | Author: nwu1015 <marchenkov3004@gmail.com>
| | Date: Sat Sep 13 12:35:57 2025 +0300
| |
| | 3
| |
* | commit a6cc3c8be62abfc88dd46e46ed8860654c01556c (br1)
|/ Author: nwu1015 <marchenkov3004@gmail.com>
| Date: Sat Sep 13 12:35:30 2025 +0300
|
| first

```
|       first
|
* commit 4cb2dfbc136b676af0e5a5df876dc1f8207cb506 (master)
  Author: nwu1015 <marchenkov3004@gmail.com>
  Date:   Sat Sep 13 12:32:52 2025 +0300
:
*   commit 72d0404bc30df926d6c6d31397075381ec775223 (HEAD -> br3)
|\  Merge: c8a667b 289fd69
| | Author: nwu1015 <marchenkov3004@gmail.com>
| | Date:   Sat Sep 13 12:41:47 2025 +0300
| |
| |     Merge branch 'br2' into br3
| |
| * commit 289fd690e58c57ab970153837972fb22d260d566 (br2)
| | Author: nwu1015 <marchenkov3004@gmail.com>
| | Date:   Sat Sep 13 12:35:49 2025 +0300
| |
| |     2
| |
* | commit c8a667b9f3f855dfa4fe53f3aeb712c741f2cc83
| | Author: nwu1015 <marchenkov3004@gmail.com>
| | Date:   Sat Sep 13 12:37:58 2025 +0300
| |
| |     A
| |
* | commit f227e5b2eb6a51545e7bef106f758a697b6c5b4b
| | Author: nwu1015 <marchenkov3004@gmail.com>
| | Date:   Sat Sep 13 12:36:36 2025 +0300
| |
| |     rebase
| |
* | commit d2e6bb3a5cbc6fdb1e1a8aeb01749da500fc23c5
| | Author: nwu1015 <marchenkov3004@gmail.com>
| | Date:   Sat Sep 13 12:35:57 2025 +0300
| |
:
*   commit 72d0404bc30df926d6c6d31397075381ec775223 (HEAD -> br3)
|\  Merge: c8a667b 289fd69
| | Author: nwu1015 <marchenkov3004@gmail.com>
| | Date:   Sat Sep 13 12:41:47 2025 +0300
```



Висновок: Під час виконання лабораторної роботи було вивчено основні команди в роботі з децентралізованими системами контролю версій на прикладі роботи з сучасною системою Git. Створено Git репозиторій, додано декілька віток, а також було вирішено конфлікт між схожими файлами в цих вітках.