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 «Теорія розробки програмного забезпечення»

Тема: «Системи контроля версій. Розподілена система контролю версій «Git»»

**Виконав:**
Студент групи ІА-34
Цап Андрій

**Перевірив:**
Мягкий Михайло Юрійович

Київ – 2025

# Зміст

# Вступ

Мета: Навчитися виконувати основні операції в роботі з децентралізованими системами контролю версій на прикладі роботи з сучасною системою Git.

У процесі командної розробки програмного забезпечення важливо забезпечити контроль над змінами у вихідному коді та можливість відстеження історії його розвитку.

Однією з найбільш поширених сучасних систем контролю версій є Git — розподілена система, що забезпечує високу швидкість роботи, гнучкість, зручну роботу з гілками та підтримку віддалених репозиторіїв.

У цій лабораторній роботі виконано створення Git-репозиторію, роботу з комітами, гілками, тегами, конфліктами злиття, а також взаємодію з віддаленим репозиторієм, що дозволило закріпити базові навички використання Git у процесі розробки.

# 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 привели до появи помилки. Таким чином можна знайти хто, коли і які зміни зробив в коді, а також чому ці зміни були зроблені.

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 ).

RCS - однією з основних нововведень RCS було використання дельт для зберігання змін (тобто зберігаються ті рядки, які змінилися, а не весь файл). Однак він мав низку недоліків.

Етап централізованих систем - на початку 90-х почалася епоха централізованих систем контролю версій. У цей період розробники почали переходити до централізованих систем, що дозволяли працювати кільком користувачам одночасно через сервер. Один із перших найпопулярніших систем (і досі використовувана) система контролю версій – CVS

SVN – у порівнянні з CVS це був наступний крок. Надійна та швидкодіюча систему контролю версій, яка зараз розробляється в рамках проєкту Apache Software Foundation. Вона реалізована за технологією клієнт-сервер та відрізняється неймовірною простотою – дві кнопки (commit, update). Порівняно з CVS, це удосконалена централізована система з кращим управлінням комітами та резервними копіями.

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.

Git - 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.

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, що найчастіше використовуються.

# Хід роботи

```
C:\Users\petro>git init lab123
Initialized empty Git repository in C:/Users/petro/lab123/.git/

C:\Users\petro>cd lab123
```

Рис 1. Створення локального репозиторію

```
C:\Users\petro\lab123>mkdir first_dir

C:\Users\petro\lab123>cd first_dir

C:\Users\petro\lab123\first_dir>echo "tst"> text.txt

C:\Users\petro\lab123\first_dir>git add text.txt

C:\Users\petro\lab123\first_dir>git commit -m "first_commit"
[master (root-commit) bc9a9a5] first_commit
 1 file changed, 1 insertion(+)
 create mode 100644 first_dir/text.txt
```

Рис 2. Створення директорії і першого файла, а також його коміт

```
C:\Users\petro\lab123\first_dir>git checkout -b first_br
Switched to a new branch 'first_br'

C:\Users\petro\lab123\first_dir>git switch -c second_br
Switched to a new branch 'second_br'
```

Рис 3. Створення 2 гілок різними способами

```
C:\Users\petro\lab123\first_dir>git branch
  first_br
  master
* second_br
```

Рис 4. Виведення всіх локальних гілок

```
C:\Users\petro\lab123\first_dir>git checkout first_br
Switched to branch 'first_br'

C:\Users\petro\lab123\first_dir>echo "зміна з first_br"> text.txt

C:\Users\petro\lab123\first_dir>git commit -am "Update text.txt in first_br"
[first_br 790e859] Update text.txt in first_br
 1 file changed, 1 insertion(+), 1 deletion(-)
```

Рис 5.Перехід і  змінення тексту в файлі на першій гілці і коміт змін

```
C:\Users\petro\lab123\first_dir>git checkout master
Switched to branch 'master'

C:\Users\petro\lab123\first_dir>echo "зміна з master"> text.txt

C:\Users\petro\lab123\first_dir>git commit -am "Update text.txt in master"
[master c6a3649] Update text.txt in master
 1 file changed, 1 insertion(+), 1 deletion(-)
```

Рис 6. Перехід і змінення тексту в файлі на другій гілці і коміт змін

```
C:\Users\petro\lab123\first_dir>git merge first_br
Auto-merging first_dir/text.txt
CONFLICT (content): Merge conflict in first_dir/text.txt
Automatic merge failed; fix conflicts and then commit the result.
```

Рис 7. Спроба злити 2 гілки і створення конфлікту злиття

```
C:\Users\petro\lab123\first_dir>type text.txt
<<<<<<< HEAD
"зм?на з master"
=======
"зм?на з first_br"
>>>>>>> first_br
```

Рис 8. Виведення вмісту файла і вручну зміна тексту в цьому файлі

```
C:\Users\petro\lab123\first_dir>git add text.txt

C:\Users\petro\lab123\first_dir>git commit -am "Resolve problem "
[master 2bd6dec] Resolve problem

C:\Users\petro\lab123\first_dir>type text.txt

"зм?на з master + зм?на з first_br"
```

Рис 9. Коміт файлу і показ вмісту після зміни

```
C:\Users\petro\lab123\first_dir>git log
commit 2bd6dec17276e1ddd14653b65f619f4ad18b1272 (HEAD -> master)
Merge: c6a3649 790e859
Author: Andrey <petrodrovosek@201005@gmail.com>
Date:   Sat Sep 20 15:56:51 2025 +0300

    Resolve problem

commit c6a3649ffdad6e7bec5bd29f45e53f6f895755c7
Author: Andrey <petrodrovosek@201005@gmail.com>
Date:   Sat Sep 20 15:54:33 2025 +0300

    Update text.txt in master

commit 790e859c1c7a90446514fc65a05a06b4ccfe777d (first_br)
Author: Andrey <petrodrovosek@201005@gmail.com>
Date:   Sat Sep 20 15:53:47 2025 +0300

    Update text.txt in first_br

commit bc9a9a551d498d723cf87e1de7f0bce35ab98fa4 (second_br)
Author: Andrey <petrodrovosek@201005@gmail.com>
Date:   Sat Sep 20 15:48:27 2025 +0300

    first_commit
```

Рис 10. Історія комітів

# Висновки

У ході виконання лабораторної роботи було вивчено основні принципи роботи з розподіленою системою контролю версій Git. Створено локальний репозиторій, виконано додавання та фіксацію змін, створення і перемикання між гілками, злиття гілок та вирішення конфлікту. Також було переглянуто історію комітів і продемонстровано ключові команди, необхідні для роботи з репозиторієм.

Отримані навички дозволяють ефективно керувати змінами в коді, працювати з паралельними гілками розробки та впевнено використовувати Git у реальних програмних проєктах.