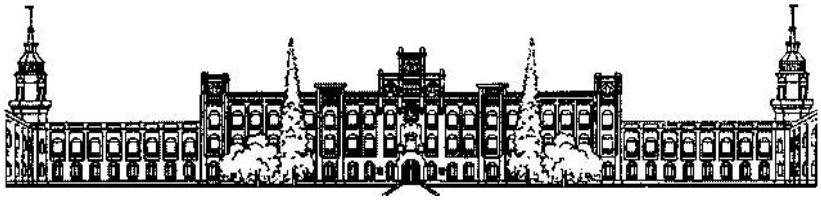

Міністерство освіти і науки України
Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”
Факультет інформатики та обчислювальної техніки
Кафедра інформатики та програмної інженерії

Лабораторна робота №1

з дисципліни «Технології розроблення програмного забезпечення»

Тема: «Дослідження способів формування нечітких множин і операцій над ними»

Виконав:

студент групи ІА-32

Нагорний Максим

Перевірив:

Мягкий М. Ю.

Київ 2025

Зміст

Вступ

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.

Завдання

- Ознайомитись із короткими теоретичними відомостями.

- Створити Git репозиторій.

- Клонувати Git репозиторій.

- Продемонструвати базову роботу з репозиторієм: створення версій, додавання тегів, робіт з гілками (створення та злиття), робота з комітами, вирішення конфліктів, а також робота з віддаленим репозиторієм.

Теоретичні відомості

2.1 Призначення систем управління версіями

У сучасній розробці програмного забезпечення контроль версій є одним із найважливіших елементів ефективної організації командної роботи.

Під час створення програмного продукту до коду вносяться численні зміни: додаються нові функції, виправляються помилки, модифікуються файли документації, конфігураційні файли тощо.

Без спеціальних інструментів складно відстежити, хто, коли і які саме зміни здійснював, а також повернутися до попереднього стану проєкту.Саме для цього і призначені системи контролю версій (VCS — Version Control Systems).

2.2 Поняття системи контролю версій

Система контролю версій — це програмний інструмент, який дозволяє зберігати історію змін у файлах, відслідковувати внесені правки, створювати паралельні гілки розробки та керувати версіями коду.Вона фіксує всі зміни у вигляді комітів (версій), що містять опис виконаних дій, автора та час внесення змін.

Основні функції систем контролю версій:

- Збереження інформації про всі редагування файлів проєкту.
- Відновлення попередніх станів файлів або всього проєкту.
- Одночасна робота декількох розробників над одним проєктом.
- Злиття різних версій одного файлу або гілки.
- Автоматичне або ручне узгодження змін при одночасному редагуванні.
- Захист від втрати даних завдяки централізованому або розподіленому збереженню.

2.3 Класифікація систем контролю версій

СКВ бувають декількох видів: локальні, централізовані, розподілені.

У локальних системах зберігаються лише на одному комп’ютері. Прикладом може бути збереження архівів файлів з різними версіями (v1, v2, final тощо). Недоліки — відсутність командної роботи, неможливість об’єднання змін різних користувачів.

У централізованих системах існує єдиний сервер, який зберігає головний репозиторій. Кожен розробник отримує робочу копію файлів, а зміни відправляє назад на сервер.

Розподілені системи на відміну від централізованих дозволяють кожному користувачу мати повну копію всього репозиторію з історією змін. Це забезпечує автономність роботи: користувач може робити коміти, створювати гілки, виконувати злиття без доступу до мережі, а потім синхронізувати зміни з іншими.

2.4 Розподілена система контролю версій Git

Git — найпопулярніша розподілена система контролю версій, створена у 2005 році Лінусом Торвальдсом для керування розробкою ядра операційної системи Linux.Відтоді Git став стандартом де-факто у сфері програмної розробки завдяки своїй швидкості, гнучкості та надійності.

Основна ідея Git, як і будь-якої іншої розподіленої системи контроля версій – кожен розробник має власний репозиторій, куди складаються зміни (версії) файлів, та синхронізація між розробниками виконується за допомогою синхронізації репозиторіїв. Відповідно, є ряд основних команд для роботи:

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

Хід виконання

1) Створити локальний репозиторій:

```
C:\Users\Max>cd Documents

C:\Users\Max\Documents>cd trpz

C:\Users\Max\Documents\trpz>git init lab
Initialized empty Git repository in C:/Users/Max/Documents/trpz/lab/.git/
```

Рисунок 1

2) Створити дві гілки та вивести їх:

```
C:\Users\Max\Documents\trpz\lab>echo test > test.txt

C:\Users\Max\Documents\trpz\lab>add .
"add" не является внутренней или внешней
командой, исполняемой программой или пакетным файлом.

C:\Users\Max\Documents\trpz\lab>git add .

C:\Users\Max\Documents\trpz\lab>git commit -m "save"
[master (root-commit) 0ce4a6c] save
 1 file changed, 1 insertion(+)
 create mode 100644 test.txt

C:\Users\Max\Documents\trpz\lab>git checkout -b one
Switched to a new branch 'one'

C:\Users\Max\Documents\trpz\lab>git switch -c two
Switched to a new branch 'two'

C:\Users\Max\Documents\trpz\lab>git branch
  master
  one
* two
```

Рисунок 2

3) Створити конфлікт при злитті гілок та вирішити його:

```
C:\Users\Max\Documents\trpz\lab>git add .

C:\Users\Max\Documents\trpz\lab>git commit -m "save"
[two 84b4018] save
 1 file changed, 1 insertion(+), 1 deletion(-)

C:\Users\Max\Documents\trpz\lab>git checkout one
Switched to branch 'one'

C:\Users\Max\Documents\trpz\lab>git add .

C:\Users\Max\Documents\trpz\lab>git commit -m "save"
[one 38b5e24] save
 1 file changed, 1 insertion(+), 1 deletion(-)

C:\Users\Max\Documents\trpz\lab>git merge two
Auto-merging test.txt
CONFLICT (content): Merge conflict in test.txt
Automatic merge failed; fix conflicts and then commit the result.
```

Рисунок 3

4) Вивести коміти графом:

```
C:\Users\Max\Documents\trpz\lab>git log --graph
*   commit 79997c22c22eb3d6bc031269886835de1a6fae6b (HEAD -> one)
|\  Merge: 38b5e24 84b4018
| | Author: Max <your_email@example.com>
| | Date:   Sat Oct 4 10:50:43 2025 +0300
| |
| |     save 2
| |
| * commit 84b40181da9d71d10c02d5996f52b2e9fc2db666 (two)
| | Author: Max <your_email@example.com>
| | Date:   Sat Oct 4 10:49:41 2025 +0300
| |
| |     save
| |
* | commit 38b5e24f39f5992da9ce3ed1276941b9e253286b
|/  Author: Max <your_email@example.com>
|   Date:   Sat Oct 4 10:50:12 2025 +0300
|
|       save
|
* commit 0ce4a6cc18c2ce05065c51d0b3038dd490f6c59f (master)
  Author: Max <your_email@example.com>
  Date:   Sat Oct 4 10:48:19 2025 +0300

      save
```

Рисунок 4

Висновок

Під час виконання лабораторної роботи ми вчилися виконувати основні операції в роботі з децентралізованими системами контролю версій на прикладі роботи з сучасною системою Git. При виконанні ми створювали локальний репозиторій, виводили гілки та вирішували конфлікти, які виникають під час злиття.