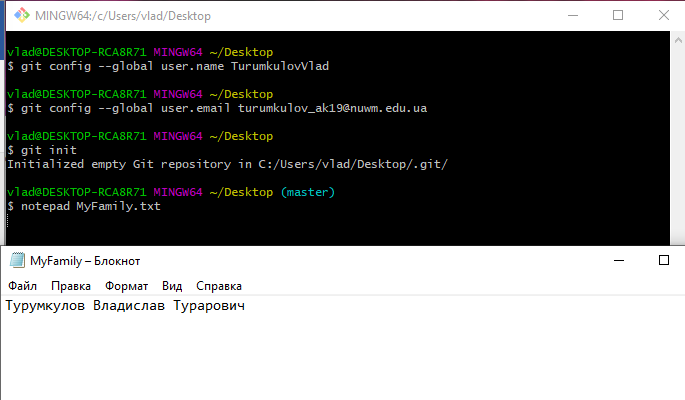
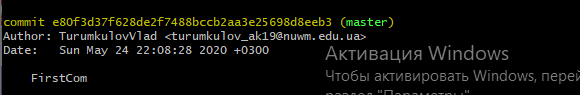
**Звіт до Лабораторної роботи №3**

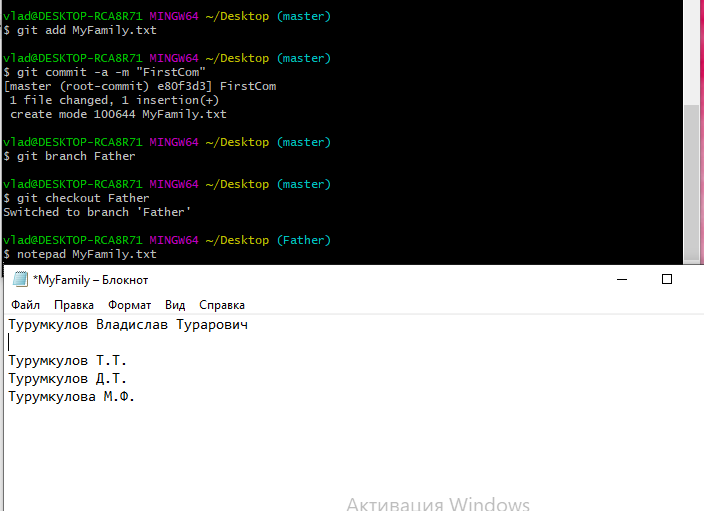
1. Встановив ім’я користувача та пароль для ідентифікації комітів. Створив в папці лабораторної локальний репозиторій. Створив текстовий файл MyFamily. Ввів в нього своє ПІБ та вивів на екран.



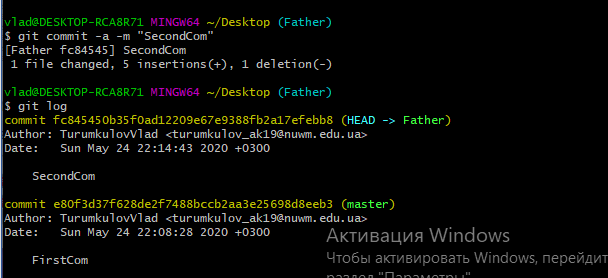
1. Створив перший коміт «FirstCom».



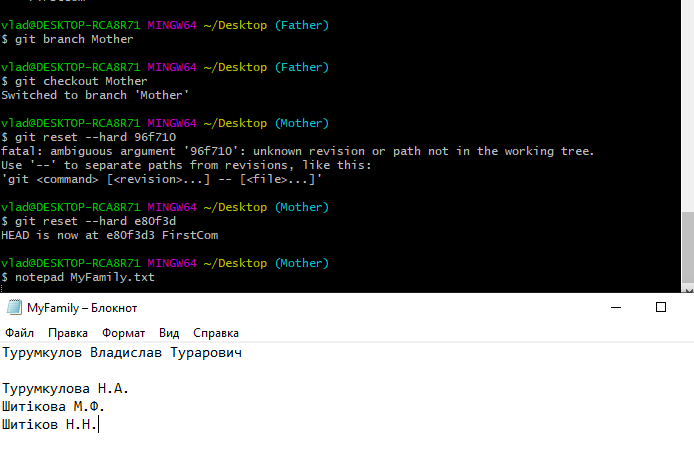
1. Створив гілку Father. Доповнив файл MyFamily ПІБ батька та його батьків. Створив другий коміт з цим файлом та запам’ятав його ідентифікатор.



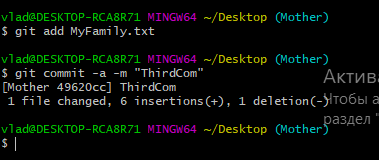
1. Ідентифікатор першого та другого коміту.



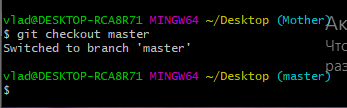
1. Створив гілку Mother. Перейшов в цій гілці до першого коміту як в індексі, так і в активній директорії. Доповнив файл MyFamily ПІБ мами та її батьків.



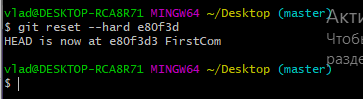
1. Створив третій коміт з цим файлом.



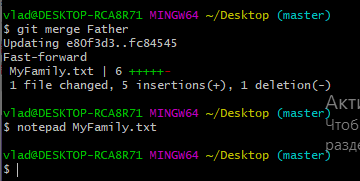
1. Повернувся в гілку master.



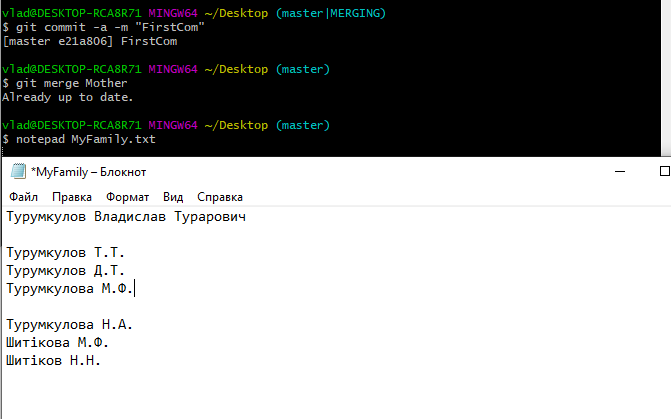
1. Перейшов в цій гілці до першого коміту як в індексі, так і в активній директорії.



1. Злив файл цього коміту з гілкою Father.



1. Злив файл цього коміту з гілкою Mother.



1. **Три основні частини технології:**

* низхідна розробка
* Наскрізний структурний контроль
* Використання базових логічних структур

: Diamond_shape_with_a_dot_inside:**Етапи створення програмного продукту: (спадна розробка 2-4):**

* Аналіз (ТЗ, можливість формалізації)
* Проектування (розробка алгоритмів)
* кодування
* тестування
* супровід
* модифікація

Використовуються алгоритми декомпозиції - розбиття задачі на підзадачі, виділені підзадачі розбиваються далі на підзадачі, формується ієрархічна структура (дані спадні, логіка висхідна, розробка спадна). Логіка піднімається на більш високий рівень. Дані на низькому рівні, на вищому логіка. Для кожної отриманої підзадачі створюємо оцінний модуль. Готуються тестуючі пакети (до етапу кодування). Підзадача не приймає рішення за модуль рівнем вище (функція відпрацювала, повернула результат, а потім аналізується). Всі дані повинні передаватися явно. Блок, функція, файл - рівні абстракції.

: Diamond_shape_with_a_dot_inside:**принципи:**

* Тести складаються до написання коду
* Підзадачі - модулі
* Не повинно бути ділянок коду, що виконують однакові дії (відсутність дублювання)
* Результат передається тільки модулю вище за ієрархією
* Обробка всіх помилок
* Глибина вкладеності обмежується цифрою 3
* Коли розбиваємо програму на підзадачі, виділяємо не більш 7 подзадач
* Коментарі - перед функціями
* Логіка - на самому верху
* Дані передаються і повертаються явно
* Рівні абстракції: файл -> функція -> блок

: Diamond_shape_with_a_dot_inside:**Три підходу програмування модулів:**

* Ієрархічний (за рівнем абстракції)
* Операційний (в порядку виклику модулів)
* Змішаний.

1. **git commit --amend**

Ця команда відкриє в текстовому редакторі повідомлення вашого останнього коммітов, для того, щоб ви могли його виправити. Коли ви збережете його і закриєте редактор, буде створений новий Комміт, що містить це повідомлення, який тепер і буде вашим останнім коммітов.

**git commit –a** автоматично викликає git add всі файли, про які відомо.

**git-commit** запис змін у сховище

1. Створити нову гілку **git branch <Назва гілки>**

Переходити на нову гілку **git checkout <назва гілки>**

Створення та автоматичний перехід в гілку - **git checkout -b <Назва гілки>**

1. Щоб видалити файл з Git, вам треба прибрати його з контрольованих файлів (вірніше, видалити його з вашого індексу) та створити коміт. Команда **git rm** це робить, а також видаляє файл з вашої робочої директорії, щоб наступного разу він не відображався неконтрольованим.
2. Після усунення конфліктів потрібно зберегти зміни **git add. git commit** та зробити **git merge**.