Київський національний університет імені Тараса Шевченка

КРЕНЕВИЧ А.П.

Методичні вказівки
до лабораторних занять із дисципліни

"Системи контролю версій"

для студентів механіко-математичного факультету

Київ – 2024

УДК 519.942+550

Рецензенти:
доктор фіз.-мат. наук, професор
доктор фіз.-мат. наук,

Рекомендовано до друку вченою радою механіко-математичного факультету
(протокол № __ від __ _____201__ року)

Креневич А.П.

Методичні вказівки до лабораторних занять із дисципліни "Системи контролю версій" для студентів механіко-математичного факультету – К.: ВПЦ "Київський Університет", 20. – ___ с.

Посібник містить перелік завдань для аудиторної та самостійної роботи з дисципліни «Системи контролю версій», що викладається студентам механіко-математичного факультету Київського національного університету імені Тараса Шевченка.

Для студентів механіко-математичного факультету та викладачів, які проводять заняття з курсу "Об’єктно-орієнтоване програмування".

ЗМІСТ

ВСТУП

ЛАБОРАТОРНА РОБОТА 1.
Поняття про системи контролю версій.

1.1. Контрольні запитання

1.1.1.

1.2. Завдання

1.2.1. Встановіть на свій персональний комп'ютер систему контролю версій Git з офіційного сайту Git. Проведіть глобальні налаштування (користувач, електронна адреса) Git.

1.2.2. Запустіть командний рядок (Git Bash, термінал, тощо). Використовуючи командний рядок

a) Створіть папку `Repository` (наприклад у кореневій папці поточного користувача) для зберігання репозиторіїв.
b) У папці `Repository` створіть папку `Test`, перейдіть у неї та створіть в ній репозиторій.
c) Додайте кілька текстових файлів з кількома рядками коду в кожному з них.
d) Перегляньте стан репозиторія командою `git status`.
e) Додайте створені файли у індекс.
f) Відредагуйте один з доданих файлів та перегляньте стан репозиторія.
g) Додайте здійснені зміни у індекс та перегляньте стан репозиторія.
h) Зробіть коміт з повідомленням «Initial commit» та перегляньте стан репозиторія.
i) Відредагуйте один або кілька з доданих файлів, перегляньте стан репозиторія та здійсніть коміт з автоматичним додаванням змін у індекс використовуючи параметр -a команди git commit.
j) Створіть новий репозиторій на GitHub та, користуючись інструкціями GitHub, надішліть у нього створений локальний репозиторій.
k) Перегляньте стан репозиторія на сторінці GitHub.

1.2.3. Встановіть на свій персональний комп'ютер одну з графічних оболонок для роботи з системою контролю версій Git (GitHub Desktop, Tortice Git, SourceTree тощо).

1.2.4. Створіть локальний репозиторій DevSW за допомогою командного рядка. У цьому репозиторії (всі операції проробіть з командного рядка)

a) Перейдіть у папку з новим репозиторієм та, використовуючи команду `git status`, перегляньте статус репозиторія.
b) Створіть перший коміт у репозиторій `git commit` з параметром -m та задайте повідомлення для коміту «`Initial commit`»
c) Додайте файл ignored.txt з текстом «Якийсь текст» у репозиторій.
d) Перегляньте статус репозиторія командою `git status`.
e) Додайте цей файл у список `gitignore`.
f) Перегляньте статус репозиторія.
g) Зробіть коміт у репозиторій з текстом «`Added gitignore`».
h) Додайте файл `main.py` в якому опишіть програму для виведення на екран повідомлення "`Hello, World`".
i) Додайте новий файл `main.py` у індекс та перегляньте статус репозиторія командою `git status`.
j) Відмініть додавання у репозиторій зазначеного вище файлу (див відповідну команду як результат виведення команди `git status`).
k) Додайте знову файл `main.py` у індекс.
l) Відредагуйте програму, створену вище, додавши функцію, що здійснює виведення повідомлення "`Hello, World`" та здійсніть виклик цієї функції 10 разів (наприклад, за допомогою циклу).
m) Перегляньте статус репозиторія командою `git status`.
n) Додайте нові зміни у індекс.
o) Перегляньте статус репозиторія.
p) Зробіть коміт у репозиторій командою `git commit` без параметрів та задайте у текстовому редакторі `vim` повідомлення для коміту «`Printing message "hello world" 10 times`».
q) Перегляньте історію комітів репозиторія командою git log.
r) Додайте у файл `main.py` нову функцію `vivaMechMat()`, що виводить повідомлення «`Viva Mech-mat faculty`» та здійсніть виклик цієї функції.
s) Зробіть коміт у репозиторій з одночасним додаванням всіх здійснених змін у індекс.
t) Перегляньте історію комітів обмеживши виведення останніми двома комітами.
u) Поправте функцію `vivaMechMat()`, так щоб виведення повідомлення закінчувалося трьома знаками оклику.
v) Виправте попередній коміт командою `git commit –amend`.

w) Перегляньте історію комітів обмеживши виведення останніми двома комітами.

1.2.5. Створіть віддалений репозиторій `DevelopSW` на сайті GitHub.

a) Склонуйте цей репозиторій собі локально на комп'ютер використовуючи командний рядок.
b) Проробіть всі операції наведені у попередній вправі для цього склонованого репозиторія.

1.2.6. Проробіть всі операції (або такі що їм відповідають) з попередніх двох вправ за допомогою будь-якої з графічних оболонок.

ЛАБОРАТОРНА РОБОТА 2.
Робота з гілками та віддаленими репозиторіями.

2.1. Контрольні запитання

2.1.1.

2.2. Завдання

2.2.1. Базові операції для роботи з гілками. Оберіть один з раніше створених тестових репозиторіїв. У випадку, якщо такі репозиторії відсутні, створіть новий репозиторій та створіть у ньому принаймні три коміти. Проробіть всі операції наведені нижче за допомогою командного рядка (Git Bash, terminal, тощо).

a) Перегляньте для цього репозиторія список гілок.
b) Перейменуйте головну гілку у гілку `main`. Якщо ваша головна гілка вже називається `main`, то перейменуйте її в `feature_main`, а потім назад на `main`.
c) Створіть гілку `develop` на поточному коміті.
d) Перейдіть на гілку `develop`.
e) Додайте на цій гілці кілька комітів.
f) Перейдіть на головну гілку `main`.
g) Проведіть зливання гілки `develop` у гілку `main`. Зверніть увагу на інформацію яку при цьому буде виведено в консоль.
h) Перегляньте список гілок та історію комітів.
i) Видаліть гілку `develop`.
j) Перегляньте поточний список гілок та історію комітів репозиторія.

2.2.2. Злиття гілок. Оберіть один з раніше створених тестових репозиторіїв.

a) Перегляньте для цього репозиторія список гілок. Перейменуйте головну гілку репозиторія у `main`.
b) Створіть гілку `develop` на поточному коміті гілки `main`.
c) Створіть файл `main_test.txt` на поточній гілці main та додайте в нього кілька рядків коду. Зробіть коміт, додавши створений файл у історію.
d) Перейдіть на гілку `develop` та додайте там файл `dev_test.txt`. Додайте в ньому кілька рядків тексту. Зробіть коміт, на цій гілці.
e) Відредагуйте файл `dev_test.txt` додавши до нього ще кілька рядків будь-якого тексту. Зробіть коміт, щоб додати ці зміни в історію.

f) Повернiться на гiлку `main`.
g) Вiдредагуйте файл `main_test.txt` додавши в нього кiлька рядкiв тексту. Зробiть комiт.
h) Проведiть злиття гiлок `main` та `develop` – у гiлку `main` влийте змiни з гiлки `develop`. Перегляньте iсторiю комiтiв. Видалiть гiлку `develop`.
i) Додайте файл common.txt та додайте в нього такий текст.

```
Hello World!
Hello Mech-Mat!
```

j) Зробiть комiт.
k) Створiть гiлку `dev` на поточному комiтi гiлки `main`.
l) Вiдредагуйте файл `common.txt` замiнивши в ньому рядок

```
Hello Mech-Mat!
```

рядком

```
Привiт Мех-мат!
```

та зробiть комiт з проведеними змiнами.
m) Повернiться на гiлку `main` та вiдредагуйте файл `common.txt` замiнивши в ньому рядок

```
Hello Mech-Mat!
```

рядком

```
Добрий день механiко-математичний!
```

та зробiть комiт з проведеними змiнами.
n) Виконайте команду злиття гiлки `dev` у головну гiлку `main`. Розв'яжiть конфлiкт, який при цьому виникне.
o) Пiсля успiшного злиття перейдiть на гiлку `dev`, вiдредагуйте файл `common.txt` (довiльним чином) зробiть комiт та поверниться назад на гiлку `main`.
p) Видалiть гiлку `dev`.

2.2.3. Робота з вiддаленими репозиторiями.
a) Створiть локально новий репозиторiй та додайте у нього принаймнi три комiти.
b) Перейменуйте головну гiлку репозиторiя у main.
c) Перегляньте список вiддалених репозиторiїв (вiн має бути порожнiм).
d) Створiть новий репозиторiй на GitHub та, користуючись iнструкцiями GitHub надiшлiть у нього створений локальний репозиторiй.
e) Перегляньте список вiддалених репозиторiїв. Перегляньте свiй репозиторiй та його гiлки на сторiнцi GitHub.

f) Створіть у поточній гілці `main` принаймні один будь-який коміт. Надішліть зміни на гілку `main_for_review` віддаленого репозиторія. Знайдіть та перегляньте зміни на сторінці GitHub для цієї гілки.
g) Використовуючи сторінку GitHub, додайте у гілки `main` та `main_for_review` принаймні по одному коміту.
h) Синхронізуйте локальний репозиторій з віддаленим репозиторієм та перегляньте історію комітів, список віддалених та локальних гілок.
i) Видаліть гілку `main_for_review` з віддаленого репозиторія користуючись командним рядком.
j) Створіть у локальному репозиторії гілку `develop`. Зробіть у ній принаймні один коміт та надішліть зміни на віддалений репозиторій.
k) Зробіть гілку develop відслідковуваною, так щоб вона автоматично синхронізувалася з гілку `develop` віддаленого репозиторія.
l) Перегляньте список відслідковуваних гілок.
m) Зробіть кілька комітів на гілці develop та надішліть їх у віддалений репозиторій.
n) Злийте гілку `develop` у головну гілку `main`. Видаліть локальну та віддалені гілки `develop`.

ЛАБОРАТОРНА РОБОТА 3.
Симуляція командної роботи.

3.1. Контрольні запитання

3.1.1.

3.2. Завдання

3.2.1. Мета цього завдання симулювати робочий процес на реальному проекті. Розділіться на команди по три – чотири людини. Одного з членів команди обирають капітаном.

a) Капітан має створити репозиторій на відділеному сервері (GitHub)
b) Капітан додає всіх учасників команди, як контрибюторів для того, щоб вони могли не лише стягувати репозиторій, але і засилати на нього свої коміти. Контрибютори мають підтвердити прийняття запрошення.
c) Капітан робить перший коміт та засилає його на віддалений репозиторій, для того щоб ініціалізувати історію.
d) Всі члени команди клонують собі цей репозиторій та розпочинають з ним працювати.
e) Для початкової ініціалізації репозиторія, капітан додає файл utils.py з кодом у якому будуть описані різні функції, додає у нього функцію для обчислення факторіалу, далі створює файл main.py у якому здійснює виклик цієї функції. Далі засилає цей доробок на віддалений репозиторій.
f) Всі члени команди оновлюють репозиторій додають по одній функції (визначення чи є число простим, обчислення НСД, визначення чи є числом степенем п'ятірки тощо) у файл utils.py та засилають ці зміни на віддалений репозиторій. *Важливо: безпосередньо перед кожним засиланням на віддалений репозиторій, необхідно стягувати зміни які там могли відбутися за час після останнього доступу до репозиторія.*
g) Після того, як всі заслали свої зміни, кожен з членів команди модифікує файл main.py, тим що додає виклик своєї функції та засилає на віддалений репозиторій.

Вказівка: У робочих проектах засилання змін на віддалений репозиторій відбувається не безпосередньо, а через маханізми рецензування коду іншими учасниками команди. На GitHub процедура рецензування відбувається через механізм, що називається Pull request – запит за зливання однієї гілки у іншу. Для використання механізму Pull Request потрібно надіслати коміт на іншу гілку віддаленого репозиторія. Для цього, після коміту на робочій гілці, виконайте команду

```
git push origin HEAD:for_review
```

На віддаленому репозиторії при цьому буде створено гілку `for_review`. Можна використовувати будь-яке інше ім'я на ваш розсуд. Після засилання відкриваємо репозиторій на GitHub, створюємо Pull Request з вашої гілки у робочу гілку (запит для зливання вашої гілки у робочу гілку). Додаєте у ролі рецензентів колег з вашої команди.

h) У цьому пункті спробуйте реалізувати засилання комітів у віддалений репозиторій на GitHub через механізм Pull Request. Проробіть операції подібні до тих, що були описані у попередніх пунктах, ще принаймні три рази (внесення зміна, коміт, стягування з репозиторія, вирішення конфліктів, заливання на віддалений репозиторій). Додайте різні функції які були написані вами у рамках попередніх курсів у файл `utils.py` та здійсніть їхні виклики. Передбачається, що кожен учасник додає різні функції.

Проробіть вище переведені операції для іншого члена команди у ролі капітана.

ЛАБОРАТОРНА РОБОТА 4.
Робота з історією комітів.

4.1. Контрольні запитання

4.1.1.

4.2. Завдання

4.2.1. Оберіть один з раніше створених тестових репозиторіїв. У випадку, якщо такі репозиторії відсутні, створіть новий репозиторій та створіть у ньому принаймні три коміти. Проробіть всі операції наведені нижче за допомогою командного рядка (Git Bash, terminal, тощо).

a) Перегляньте історію цього репозиторія.
b) Створіть текстовий файл `my_file.txt` та додайте в нього текст
```
1: Hello, world!
2: Hello, world!
3: Hello, world!
```
c) Додайте файл до індексу та створіть коміт.
d) Надішліть зміни до віддаленого репозиторія (якщо віддалений репозиторій для цього репозиторія відсутній – створіть його на ресурсі GitHub).
e) Змініть файл my_file.txt додавши до нього рядок
```
4: Hello, world!
```
f) Додайте зміни у індекс. Після цього додайте ще кілька рядків.
```
5: Hello, world!
6: Hello, world!
```
g) Перегляньте статус вашого репозиторія.
h) Приховайте зміни командою `stash` та перегляньте статус вашого репозиторія.
i) Відновіть приховані зміни та перегляньте статус репозиторія.
j) Відмініть зміни зроблені (команда restore) – спочатку з індексу, потім з робочої директорії. Перегляньте статус репозиторія.
k) Змініть файл `my_file.txt` додавши до нього рядок
```
7: Hello, world!
```
l) Змініть файл `my_file.txt` на віддаленому репозиторії безпосередньо на сторінці GitHub, вставивши на початку рядок
```
0: Hello, world! – from remote
```
та зробіть там коміт
m) Перегляньте статус локального репозиторія.

n) Спробуйте стягнути зміни з віддаленого репозиторія та вивчіть відповідь, яку поверне Git.
o) Приховайте зміни у локальному репозиторії командою `stash` та повторіть спробу стягнути зміни з віддаленого репозиторія.
p) Відновіть приховані зміни (вирішіть конфлікт, якщо він виявиться).
q) Перегляньте файл `my_file.txt`, переконайтеся, що всі зміни (з віддаленого репозиторія та зі стешу) коректно застосувалися.
r) створіть коміт та відправте зміни у віддалений репозиторій.

4.2.2. Оберіть один з раніше створених тестових репозиторіїв. У випадку, якщо такі репозиторії відсутні, створіть новий репозиторій та створіть у ньому принаймні три коміти. Проробіть всі операції наведені нижче за допомогою командного рядка (Git Bash, terminal, тощо).

a) Перегляньте історію цього репозиторія. Перегляньте список гілок цього репозиторія. Перейдіть на головну гілку репозиторія. Перейменуйте її в `main`, якщо вона має іншу назву.
b) Перейдіть на головну гілку репозиторія `main`. Якщо головна гілка має іншу назву – перейменуйте її у `main`.
c) Створіть файл `utils.py`, що містить функції для обчислення факторіалу, визначення чи є число простим.
d) Створіть файл main.py, у якому здійсніть виклик однієї з функцій, що міститься у файлі `utils.py`.
e) Створіть коміт для збереження зроблених вище змін.
f) Створіть гілку dev5 та перейдіть на неї.
g) Додайте у файл `utils.py` функцію, що визначає чи є число степенем 5. Та здійсніть виклик цієї функції з головного файлу програми main.py.
h) Закомітьте зміни на гілці `dev5`.
i) Додайте у файл `utils.py` функцію, що визначає чи є число степенем 2. Та здійсніть виклик цієї функції з головного файлу програми `main.py`.
j) Без створення коміту, перенесіть зміни отримані на попередньому кроці на головну гілку `main` – для цього приховайте зміни командою stash, перейдіть на гілку main та відновіть приховані зміни.
k) Зробіть коміт для отриманих змін.
l) Об’єднайте гілки main та `dev5` – вирішіть конфлікти за необхідності.
m) Видаліть гілку dev5 після об’єднання.

4.2.3. Рух по історії комітів – checkout.

a) Створіть новий репозиторій та перейменуйте головну гілку на main.
b) Створіть у ньому 5 комітів з довільними змінами з повідомленнями
```
commit 01 – initial
commit 02
…
commit 05
```
c) Виведіть історію комітів на зверніть увагу на хеш значення комітів.
d) Перегляньте зміни зроблені на кількох комітах, наприклад на коміті «commit 02», «commit 04»та «commit 05».
e) Переключіться командою checkout на коміт «commit 03», використовуючи хеш цього коміту.
f) Створіть гілку tmp на поточному коміті.
g) Додайте тестовий файл commit03.txt з довільним текстом, додайте його в індекс та створіть коміт.
h) Додайте ще кілька рядків у цей файл.
i) Без створення нового коміту спробуйте переключитися командою checkout на гілку main. Проаналізуйте повідомлення, що поверне Git.
j) Відмініть зміни зроблені (команда restore).
k) Переключіться на гілку main .
l) Злийте гілку tmp у поточну гілку main та видаліть гілку tmp.

4.2.4. Рух по історії комітів – reset. Виконайте це завдання у репозиторії створеному у рамках попередньої задачі.

a) Перейдіть на головну гілку main репозиторія. Головна гілка має вказувати на коміт з коміт-повідомленням «commit 05».
b) Створіть файл main.txt та додайте в нього текст
```
1: Hello, world!
2: Hello, world!
```
c) Створіть коміт з повідомленням «commit 06 - added main.txt». Збережіть хеш цього коміту.
d) Додайте ще один рядок тексту у файл main.txt
```
3: Hello, world!
```
e) Додайте зміни у індекс.
f) Додайте ще один рядок тексту у файл main.txt
```
4: Hello, world!
```
g) Переключіться командою reset з параметром --hard на попередній коміт «commit 05» використовуючи вказівник HEAD.

h) Перегляньте стан робочої директорії, індексу та історії комітів. Зверніть увагу, що всі зміни, що були пророблені на кроках b) – f) тепер втрачені.
i) Повторіть операції пророблені на кроках b) – f).
j) Переключіться командою `reset` з параметром `--soft` на попередній коміт «`commit 05`» використовуючи вказівник HEAD.
k) Перегляньте стан робочої директорії, індексу та історії комітів. Які висновки можна зробити про стан робочої директорії, індексу та історії комітів?

4.2.5. Зливання набору комітів в один (reset).

a) Створіть новий репозиторій та перейменуйте головну гілку на `main`.
b) Створіть у ньому файл `my_file.txt` з текстом
```
1: Hello, world!
```
c) Збережіть ці зміни у коміті з коміт-повідомленням «`commit 01`».
d) Додайте ще 4 коміти з коміт-повідомленнями
```
commit 02
commit 03
commit 04
commit 05
```
у кожному з яких до раніше створеного файлу додайте по відповідному рядку тексту
```
2: Hello, world!
3: Hello, world!
4: Hello, world!
5: Hello, world!
```
e) Переключіться за допомогою команди `reset` з параметром `--soft` (або `--mixed`) на найперший коміт «`commit 01`».
f) Перегляньте стан робочої директорії, індексу та історії комітів. Переконайтеся, що історія комітів мітить лише перший коміт, проте зміни зроблені в інших чотирьох комітах містяться у робочій директорії.
g) Додайте ці зміни в індекс.
Вказівка: якщо ви використали команду `reset` *з параметром* `--soft` *то цей крок не потрібно буде здійснювати, адже* команда `reset --soft` *не змінила індексу. Якщо ви використали команду* `reset` *з параметром* `--mixed`*, то індекс буде змінений відповідно до коміту на який було здійснено перехід.*
h) та створіть коміт за параметром `--amend`.

і) Перегляньте історію комітів. Переконайтеся у тому, що всі зміни, що попередньо містилися у п’яти комітах тепер містяться у коміті єдиному «`commit 01`».

СПИСОК ЛІТЕРАТУРИ ТА ДОДАТКОВИХ ДЖЕРЕЛ

1.