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.kt в якому опишіть програму для виведення на екран повідомлення “Hello, World”.
i) Додайте новий файл main.kt 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.kt 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.kt 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) Поверніться на гілку main.

g) Відредагуйте файл main_test.txt додавши в нього кілька рядків тексту. Зробіть коміт.
h) Проведіть злиття гілок main та develop – у гілку main влийте зміни з гілки develop. Перегляньте історію комітів. Видаліть гілку develop.
i) Додайте файл common.txt та додайте в нього такий текст.

Hello World!
Hello Mech-Mat!

j) Зробіть коміт.
k) Створіть гілку dev на поточному коміті гілки main.
l) Відредагуйте файл common.txt замінивши в ньому рядок

Hello Mech-Mat!

рядком

Привіт Мех-мат!

та зробіть коміт з проведеними змінами.
m) Поверніться на гілку main та відредагуйте файл common.txt замінивши в ньому рядок

Hello Mech-Mat!

рядком

Добрий день механіко-математичний!

та зробіть коміт з проведеними змінами.
n) Виконайте команду злиття гілки dev у головну гілку main. Розв'яжіть конфлікт, який при цьому виникне.
o) Після успішного злиття перейдіть на гілку dev, відредагуйте файл common.txt (довільним чином) зробіть коміт та поверніться назад на гілку main.
p) Видаліть гілку dev.

2.2.3. Робота з віддаленими репозиторіями.

a) Створіть локально новий репозиторій та додайте у нього принаймні три коміти.
b) Перейменуйте головну гілку репозиторія у main.
c) Перегляньте список віддалених репозиторіїв (він має бути порожнім).
d) Створіть новий репозиторій на GitHub та, користуючись інструкціями GitHub надішліть у нього створений локальний репозиторій.
e) Перегляньте список віддалених репозиторіїв. Перегляньте свій репозиторій та його гілки на сторінці 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.

СПИСОК ЛІТЕРАТУРИ ТА ДОДАТКОВИХ ДЖЕРЕЛ

1.