МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯРОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ

ВЫСШЕГО ОБРАЗОВАНИЯ

«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»ИНСТИТУТ ЦИФРОВОГО РАЗВИТИЯ

Отчет о лабораторной работе №1.3 по дисциплине основы программной инженерии

Выполнила:
Лобанова Елизавета Евгеньевна,
2 курс, группа ПИЖ-б-о-20-1

Проверил:
Доцент кафедры прикладной математики и компьютерной безопасности,
Воронкин Р.А.

Отчет защищен с оценкой_____Дата защиты_____

Ставрополь, 2021 г.

Ход работы

```
C:\Users\Acer\OneDrive\Рабочий стол\lr_opi\lr_3l>git add 1.txt

C:\Users\Acer\OneDrive\Рабочий стол\lr_opi\lr_3l>git commit -m "add 1.txt file"
[main 7c23e39] add 1.txt file
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 1.txt

C:\Users\Acer\OneDrive\Рабочий стол\lr_opi\lr_3l>git add 2.txt

C:\Users\Acer\OneDrive\Рабочий стол\lr_opi\lr_3l>git add 3.txt

C:\Users\Acer\OneDrive\Рабочий стол\lr_opi\lr_3l>git commit --amend -m "add 2.txt and 3.txt"
[main d227921] add 2.txt and 3.txt
 Date: Sat Dec 18 01:03:01 2021 +0300
 3 files changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 1.txt
 create mode 100644 2.txt
 create mode 100644 3.txt
```

1 – Индексирование файлов 1.txt, 2.txt, 3.txt и его коммиты

```
C:\Users\Acer\OneDrive\Рабочий стол\lr_opi\lr_3l>git branch my_first_branch
```

2 – Создание ветки

```
C:\Users\Acer\OneDrive\Рабочий стол\lr_opi\lr_3l>git checkout my_first_branch
Switched to branch 'my_first_branch'
```

3 – Переход на ветку

```
C:\Users\Acer\OneDrive\Рабочий стол\lr_opi\lr_3l>git add in_branch.txt

C:\Users\Acer\OneDrive\Рабочий стол\lr_opi\lr_3l>git commit -m "add in_branch.txt"
[my_first_branch bd82dbd] add in_branch.txt
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 in_branch.txt
```

4 – Коммит изменений

```
C:\Users\Acer\OneDrive\Рабочий стол\lr_opi\lr_3l>git checkout -b new_branch
Switched to a new branch 'new_branch'
```

5 – Создание и переход на ветку

```
C:\Users\Acer\OneDrive\Рабочий стол\lr_opi\lr_3l>git status
On branch new_branch
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   1.txt

no changes added to commit (use "git add" and/or "git commit -a")

C:\Users\Acer\OneDrive\Рабочий стол\lr_opi\lr_3l>git add 1.txt

C:\Users\Acer\OneDrive\Рабочий стол\lr_opi\lr_3l>git commit -m "new row in 1.txt"
[new_branch c09efc0] new row in 1.txt
 1 file changed, 1 insertion(+)
```

6 – Коммит изменений

```
C:\Users\Acer\OneDrive\Рабочий стол\lr_opi\lr_3l>git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

C:\Users\Acer\OneDrive\Рабочий стол\lr_opi\lr_3l>git merge my_first_branch
Updating d227921..1f55143
Fast-forward
 in_branch.txt | 0
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 in_branch.txt

C:\Users\Acer\OneDrive\Рабочий стол\lr_opi\lr_3l>git merge new_branch
Merge made by the 'recursive' strategy.
 1.txt | 1 +
 1 file changed, 1 insertion(+)
```

7 – Переход на ветку main и merge файлов

```
C:\Users\Acer\OneDrive\Рабочий стол\lr_opi\lr_3l>git branch -d my_first_branch
Deleted branch my_first_branch (was 1f55143).

C:\Users\Acer\OneDrive\Рабочий стол\lr_opi\lr_3l>git branch -d new_branch
Deleted branch new_branch (was c09efc0).
```

8 – Удаление веток my_first_branch и new_branch

```
C:\Users\Acer\OneDrive\Рабочий стол\lr_opi\lr_3l>git branch branch_1

C:\Users\Acer\OneDrive\Рабочий стол\lr_opi\lr_3l>git branch branch_2

C:\Users\Acer\OneDrive\Рабочий стол\lr_opi\lr_3l>git checkout branch_1
Switched to branch 'branch_1'
```

9 – Создание веток и переход на ветку branch_1

```
C:\Users\Acer\OneDrive\Рабочий стол\lr_opi\lr_3l>git add .

C:\Users\Acer\OneDrive\Рабочий стол\lr_opi\lr_3l>git commit -m "fix 1.txt and 3.txt"
[branch_1 f258f31] fix 1.txt and 3.txt
 2 files changed, 2 insertions(+), 4 deletions(-)
```

10 – Коммит изменений

```
C:\Users\Acer\OneDrive\Рабочий стол\lr_opi\lr_3l>git checkout branch_2
Switched to branch 'branch_2'

C:\Users\Acer\OneDrive\Рабочий стол\lr_opi\lr_3l>git add .

C:\Users\Acer\OneDrive\Рабочий стол\lr_opi\lr_3l>git commit -m "My fix 1.txt and 3.txt"
[branch_2 ea21983] My fix 1.txt and 3.txt
 2 files changed, 2 insertions(+), 4 deletions(-)
```

11 – Переход на ветку branch_2

```
C:\Users\Acer\OneDrive\Рабочий стол\lr_opi\lr_3l>git checkout branch_1
Switched to branch 'branch_1'

C:\Users\Acer\OneDrive\Рабочий стол\lr_opi\lr_3l>git merge branch_2
Auto-merging 3.txt
CONFLICT (content): Merge conflict in 3.txt
Auto-merging 1.txt
CONFLICT (content): Merge conflict in 1.txt
Automatic merge failed; fix conflicts and then commit the result.
```

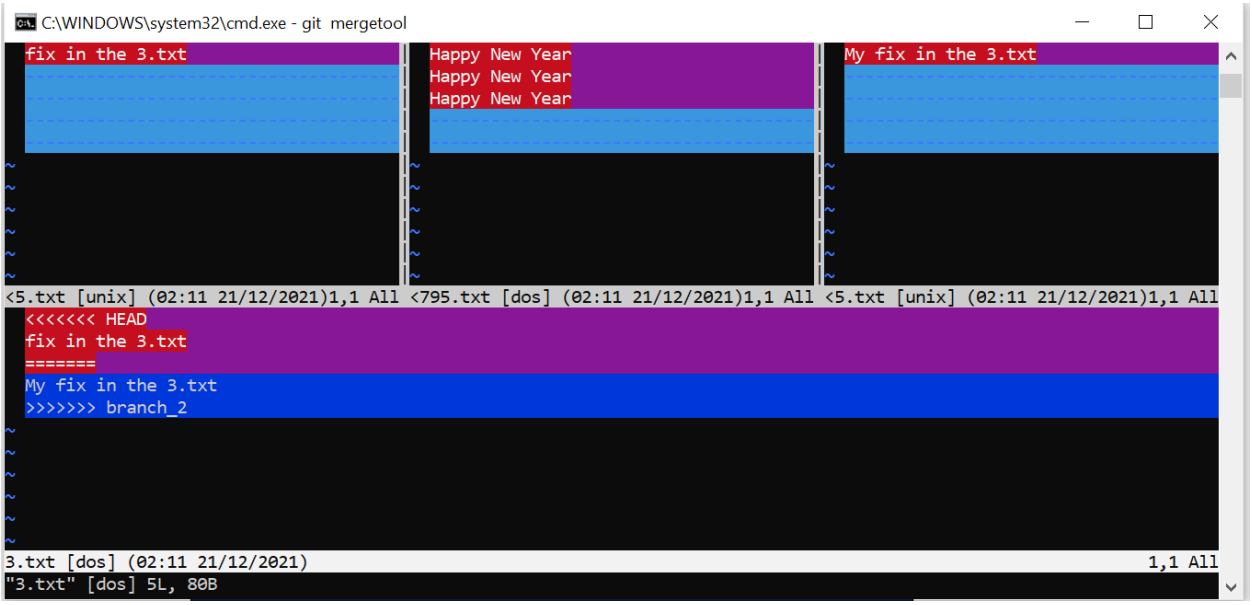
12 – Переход на ветку branch_1 и merge

```
<<<<<<< HEAD
fix in the 1.txt
=======
My fix in the 1.txt
>>>>>>> branch_2
```

13 – Решение конфликта в ручном режиме

```
C:\Users\Acer\OneDrive\Рабочий стол\lr_opi\lr_3l>git mergetool

This message is displayed because 'merge.tool' is not configured.
See 'git mergetool --tool-help' or 'git help config' for more details.
'git mergetool' will now attempt to use one of the following tools:
tortoisemerge emerge vimdiff nvimdiff
Merging:
3.txt

Normal merge conflict for '3.txt':
  {local}: modified file
  {remote}: modified file
Hit return to start merge resolution tool (vimdiff):
```

14 – Решение конфликта с помощью mergetool



15 – Решение конфликта с помощью mergetool

```
C:\Users\Acer\OneDrive\Рабочий стол\lr_opi\lr_3l>git commit -m "Commit merge"
[branch_1 2619637] Commit merge
```

16 – Коммит

```
C:\Users\Acer\OneDrive\Рабочий стол\lr_opi\lr_3l>git push origin branch_1
Enumerating objects: 26, done.
Counting objects: 100% (26/26), done.
Delta compression using up to 8 threads
Compressing objects: 100% (15/15), done.
Writing objects: 100% (25/25), 1.88 KiB | 320.00 KiB/s, done.
Total 25 (delta 8), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (8/8), done.
remote:
remote: Create a pull request for 'branch_1' on GitHub by visiting:
remote:      https://github.com/lilobanova/lr_3l/pull/new/branch_1
remote:
To https://github.com/lilobanova/lr_3l.git
 * [new branch]      branch_1 -> branch_1
```

17 – Отправка ветки branch_1

```
C:\Users\Acer\OneDrive\Рабочий стол\lr_opi\lr_3l>git fetch --all
Fetching origin
From https://github.com/lilobanova/lr_3l
 * [new branch]      branch_3   -> origin/branch_3

C:\Users\Acer\OneDrive\Рабочий стол\lr_opi\lr_3l>git checkout branch_3
Switched to a new branch 'branch_3'
Branch 'branch_3' set up to track remote branch 'branch_3' from 'origin'.
```

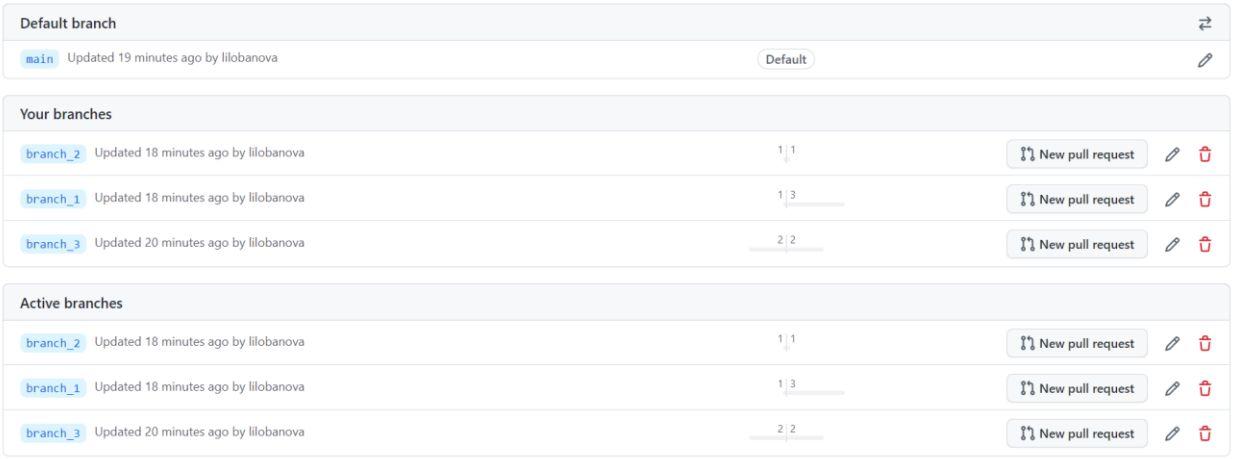
18 – Создание ветки отслеживания и переход на нее

```
C:\Users\Acer\OneDrive\Рабочий стол\lr_opi\lr_3l>git checkout branch_3
Already on 'branch_3'
Your branch is up to date with 'origin/branch_3'.

C:\Users\Acer\OneDrive\Рабочий стол\lr_opi\lr_3l>get add .
"get" не является внутренней или внешней
командой, исполняемой программой или пакетным файлом.

C:\Users\Acer\OneDrive\Рабочий стол\lr_opi\lr_3l>git add .

C:\Users\Acer\OneDrive\Рабочий стол\lr_opi\lr_3l>git commit -m "2.txt on branch_3"
[branch_3 db75bb4] 2.txt on branch_3
 6 files changed, 6 insertions(+), 2 deletions(-)
 create mode 100644 .3_BASE_795.txt.swp
 create mode 100644 .3_LOCAL_276.txt.swp
 create mode 100644 .3_LOCAL_795.txt.swp
 create mode 100644 .3_REMOTE_795.txt.swp
 create mode 100644 3_BACKUP_276.txt
```

19 – Коммит файлов на ветке branch_3

```
C:\Users\Acer\OneDrive\Рабочий стол\lr_opi\lr_3l>git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

C:\Users\Acer\OneDrive\Рабочий стол\lr_opi\lr_3l>git rebase branch_2
Successfully rebased and updated refs/heads/main.

C:\Users\Acer\OneDrive\Рабочий стол\lr_opi\lr_3l>git push origin branch_2
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'branch_2' on GitHub by visiting:
remote:      https://github.com/lilobanova/lr_3l/pull/new/branch_2
remote:
To https://github.com/lilobanova/lr_3l.git
 * [new branch]      branch_2 -> branch_2

C:\Users\Acer\OneDrive\Рабочий стол\lr_opi\lr_3l>git push origin main
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/lilobanova/lr_3l.git
   f88ecc0..ea21983  main -> main

C:\Users\Acer\OneDrive\Рабочий стол\lr_opi\lr_3l>git push origin branch_3
Enumerating objects: 10, done.
Counting objects: 100% (10/10), done.
Delta compression using up to 8 threads
Compressing objects: 100% (7/7), done.
Writing objects: 100% (8/8), 1.20 KiB | 409.00 KiB/s, done.
Total 8 (delta 3), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (3/3), done.
To https://github.com/lilobanova/lr_3l.git
   f88ecc0..db75bb4  branch_3 -> branch_3
```

20 – Отправка изменений на удаленный репозиторий



21 – проверка изменений на GitHub

Ответы на вопросы:

1. Что такое ветка?

Ветка в Git — это простой перемещаемый указатель на один из таких коммитов. Коммит – содержит метаданные и указатель на объект дерева каталогов.

2. Что такое HEAD?

HEAD – это указатель, задача которого ссылаться на определенный коммит в репозитории. HEAD – это указатель на коммит в вашем репозитории, который станет родителем следующего коммита. HEAD указывает на коммит, относительного которого будет создана рабочая копия вовремя операции checkout.

3. Способы создания веток.

Первый вариант – командой git branch Второй вариант – используя GitHub

4. Как узнать текущую ветку?

Прописав команду git branch, звездочкой будет отмечена текущая ветка.

5. Как переключаться между ветками?

Используя команду git checkout < branch_name>

6. Что такое удаленная ветка?

Удалённые ссылки — это ссылки (указатели) в ваших удалённых репозиториях, включая ветки, теги и так далее.

7. Что такое ветка отслеживания?

Ветки слежения — это ссылки на определённое состояние удалённых веток.

8. Как создать ветку отслеживания?

Командой – git checkout --track / Или использовав команды git fetch git checkout Или создать ветку с названием отличным от данной git checkout -b /

9. Как отправить изменения из локальной ветки в удаленную ветку?

git push

10. В чем отличие команд git fetch и git pull?

git fetch – просто получает с сервера все изменения, которых у вас ещё нет, но не будет изменять состояние вашей рабочей директории. git pull – является командой git fetch с тем отличем, что после git fetch происходит команда git merge.

11. Как удалить локальную и удаленную ветки?

Для удаления локальной ветки необходимо написать команду git branch –d , если необходимо удалить ветку с несохраненными изменениями необходимо прописать команду git branch –D . Для удаления удаленной ветки необходимо прописать команду git push –delete

12. Изучить модель ветвления git-flow (использовать материалы статей https://www.atlassian.com/ru/git/tutorials/comparingworkflows/gitflow-orkflow, https://habr.com/ru/post/106912/). Какие основные типы веток присутствуют в модели git-flow? Как организована работа с ветками в модели git-flow? В чем недостатки git-flow?

В нём присутствуют такие ветки как «feature», «release» и «hotfix» прописать команду «git flow init»

git flow feature start MYFEATURE

git flow release start RELEASE [BASE]

git flow release publish RELEASE

git flow release track RELEASE

git flow release finish RELEASE

git flow hotfix start VERSION [BASENAME]

git flow hotfix finish VERSION

1. Git Flow может замедлять работу, когда приходится ревьювить большие пулл реквесты, когда вы пытаетесь выполнить итерацию быстро.

2. Релизы сложно делать чаще, чем раз в неделю.

3. Большие функции могут потратить дни на мерж и резолв конфликтов и форсировать несколько циклов тестирования.

4. История проекта в гите имеет кучу merge commits и затрудняет просмотр реальной работы.

5. Может быть проблематичным в CI/CD сценариях.