

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное
образовательное учреждение высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»**

Кафедра инфокоммуникаций

**Основы кроссплатформенного программирования
Отчет по лабораторной работе №2-1**

Работа с функциями в языке Python3

Выполнила студентка группы
ИТС-б-о-20-1 (2)

Швецова К.С. « » _____ 20__ г.

Подпись студента _____

Работа защищена « » _____ 20__ г.

Проверил к.т.н., доцент

Кафедры инфокоммуникаций

Воронкин Р.А.

(подпись)

Ставрополь 2021

Ссылка на репозиторий – <https://github.com/ShveczovaKS/1lab2k>

Цель работы: исследование базовых возможностей по работе с локальными и удаленными ветками Git.

Ход работы:

1. Создала общедоступный репозиторий на GitHub, в котором будет использована лицензия MIT:

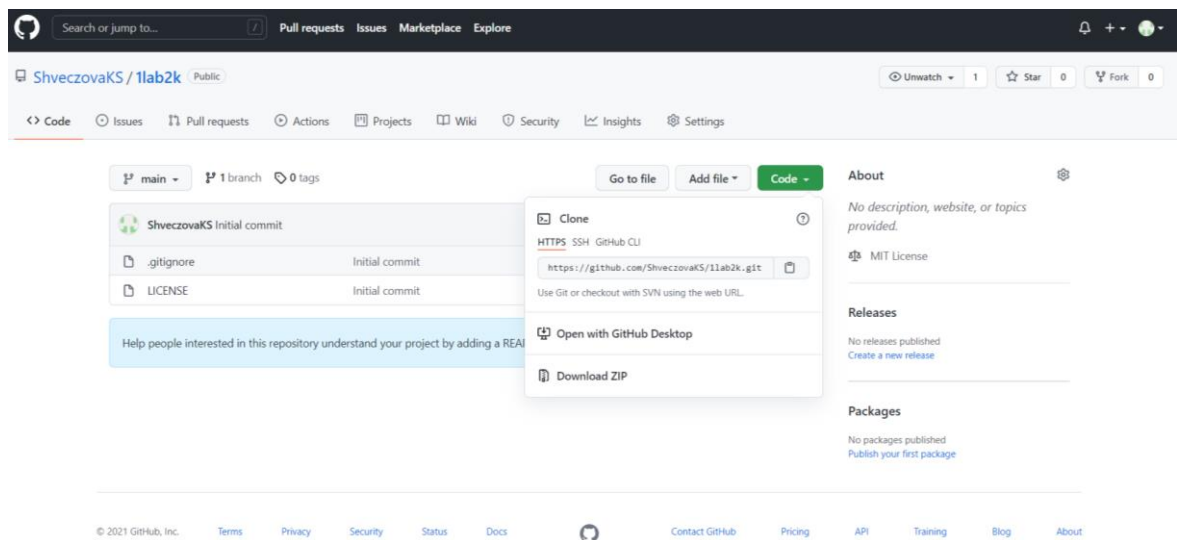


Рисунок 1. Создание репозитория

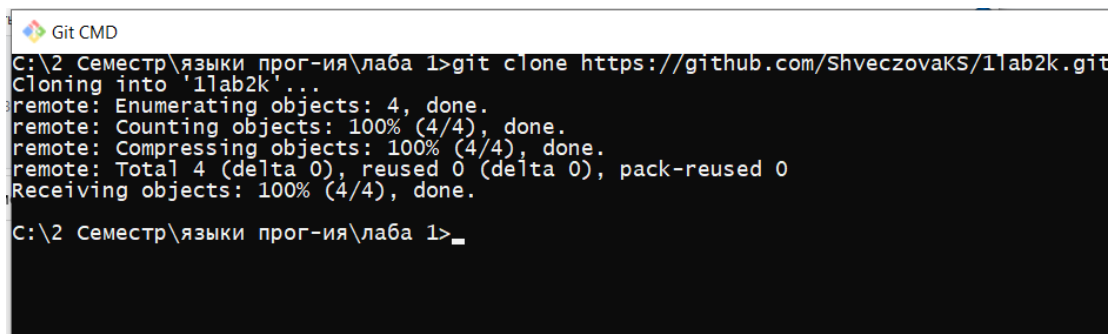


Рисунок 2. Клонирование репозитория

2. Создала три файла: 1.txt, 2.txt, 3.txt:

прог-ия > лаба 1 > 1lab2k

Имя

.gitignore


1.txt

2.txt

3.txt

Рисунок 3. Созданные текстовые документы


3. Проиндексировала первый файл и сделала коммит с комментарием "add 1.txt file":

 Git CMD

```
C:\2 Семестр\языки прог-ия\лаба 1\1lab2k>git add 1.txt
C:\2 Семестр\языки прог-ия\лаба 1\1lab2k>git commit -m "Добавила 1.txt"
[main 0206f19] Добавила 1.txt
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 1.txt
C:\2 Семестр\языки прог-ия\лаба 1\1lab2k>_
```

Рисунок 4. Индексация первого файла


4. Проиндексировала второй и третий файлы и сделала коммит с новым комментарием "add 2.txt and 3.txt."

 Git CMD

```
create mode 100644 1.txt
C:\2 Семестр\языки прог-ия\лаба 1\1lab2k>git add 2.txt
C:\2 Семестр\языки прог-ия\лаба 1\1lab2k>git add 3.txt
C:\2 Семестр\языки прог-ия\лаба 1\1lab2k>git commit -m "Добавила 2.txt и 3.txt"
[main 4a4608a] добавила 2.txt и 3.txt
2 files changed, 0 insertions(+), 0 deletions(-)
create mode 100644 2.txt
create mode 100644 3.txt
C:\2 Семестр\языки прог-ия\лаба 1\1lab2k>
```

Рисунок 5. Индексация второго и третьего файлов

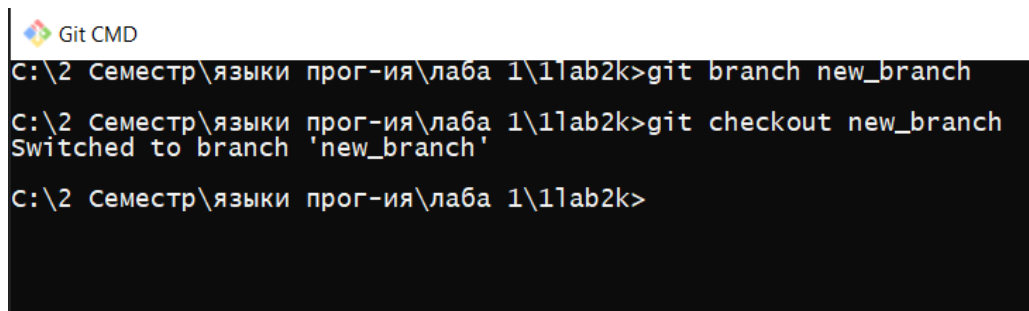
5. Создала новую ветку my_first_branch, перешла на нее и создала новый файл in_branch.txt, закомитив изменения:

 Git CMD

```
C:\2 Семестр\языки прог-ия\лаба 1\1lab2k>git branch my_first_branch
C:\2 Семестр\языки прог-ия\лаба 1\1lab2k>git checkout me_first_branch
error: pathspec 'me_first_branch' did not match any file(s) known to git
C:\2 Семестр\языки прог-ия\лаба 1\1lab2k>git checkout my_first_branch
Switched to branch 'my_first_branch'
C:\2 Семестр\языки прог-ия\лаба 1\1lab2k>git add in_branch.txt
C:\2 Семестр\языки прог-ия\лаба 1\1lab2k>git commit -m "Добавила in_branch.txt"
[my_first_branch 0c5ba8d] Добавила in_branch.txt
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 in_branch.txt
C:\2 Семестр\языки прог-ия\лаба 1\1lab2k>
```

Рисунок 6. Создание новой ветки и коммит изменений

6. Создала и сразу перешла на ветку new_branch:



```
Git CMD
C:\2 Семестр\языки прог-ия\лаба 1\1lab2k>git branch new_branch
C:\2 Семестр\языки прог-ия\лаба 1\1lab2k>git checkout new_branch
Switched to branch 'new_branch'
C:\2 Семестр\языки прог-ия\лаба 1\1lab2k>
```

Рисунок 7. Создание новой ветки

7. Сделала изменения в файле 1.txt, добавила строчку “new row in the 1.txt file”, закоммитив изменения:

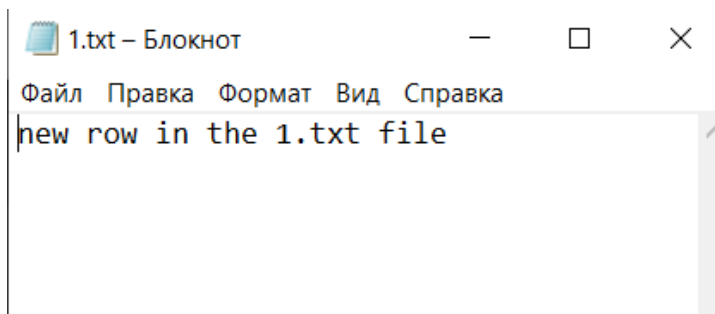
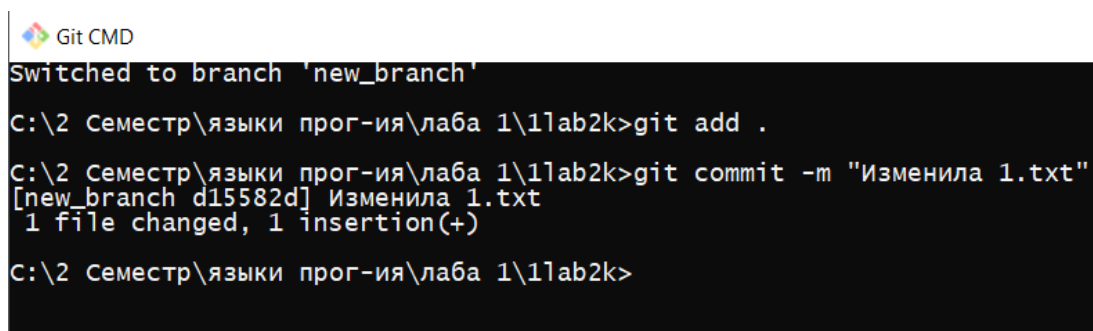


Рисунок 8. Изменение текстового файла



```
Git CMD
Switched to branch 'new_branch'
C:\2 Семестр\языки прог-ия\лаба 1\1lab2k>git add .
C:\2 Семестр\языки прог-ия\лаба 1\1lab2k>git commit -m "Изменила 1.txt"
[new_branch d15582d] Изменила 1.txt
1 file changed, 1 insertion(+)
C:\2 Семестр\языки прог-ия\лаба 1\1lab2k>
```

Рисунок 9. Коммит изменений

8. Перешла на ветку master и слила ветки master и my_first_branch, после чего слила ветки master и new_branch:

```

Git CMD
error: pathspec 'master' did not match any file(s) known to git
C:\2 Семестр\языки прог-ия\лаба 1\1lab2k>git checkout main
Switched to branch 'main'
Your branch is ahead of 'origin/main' by 2 commits.
(use "git push" to publish your local commits)

C:\2 Семестр\языки прог-ия\лаба 1\1lab2k>git merge my_first_branch
Updating 4a4608a..0c5ba8d
Fast-forward
 in_branch.txt | 0
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 in_branch.txt

C:\2 Семестр\языки прог-ия\лаба 1\1lab2k>git merge new_branch
Updating 0c5ba8d..d15582d
Fast-forward
 1.txt | 1 +
 1 file changed, 1 insertion(+)

C:\2 Семестр\языки прог-ия\лаба 1\1lab2k>git branch
* main
  my_first_branch
  new_branch

C:\2 Семестр\языки прог-ия\лаба 1\1lab2k>_

```

Рисунок 10. Слияние веток

9. Удалить ветки my_first_branch и new_branch и создала ветки branch_1 и branch_2:

```

Git CMD
C:\2 Семестр\языки прог-ия\лаба 1\1lab2k>git branch
* main
  my_first_branch
  new_branch

C:\2 Семестр\языки прог-ия\лаба 1\1lab2k>git branch branch_1

C:\2 Семестр\языки прог-ия\лаба 1\1lab2k>git branch branch_2

C:\2 Семестр\языки прог-ия\лаба 1\1lab2k>git branch -d my_first_branch
Deleted branch my_first_branch (was 0c5ba8d).

C:\2 Семестр\языки прог-ия\лаба 1\1lab2k>git branch -d new_branch
Deleted branch new_branch (was d15582d).

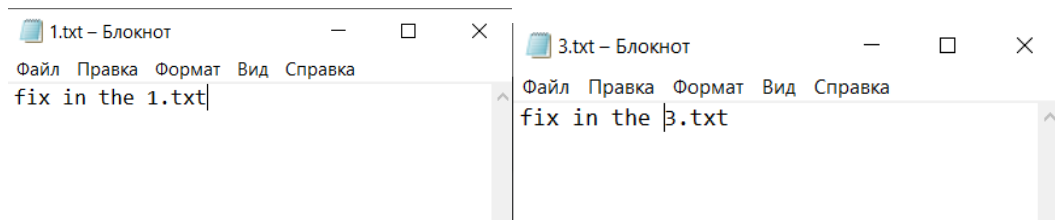
C:\2 Семестр\языки прог-ия\лаба 1\1lab2k>git branch
  branch_1
  branch_2
* main

C:\2 Семестр\языки прог-ия\лаба 1\1lab2k>_

```

Рисунок 11. Удаление и создание новых веток

10. Перешла на ветку branch_1 и изменила файл 1.txt на текст “fix in the 1.txt”, изменила файл 3.txt на текст “fix in the 3.txt”, закоммитив изменения

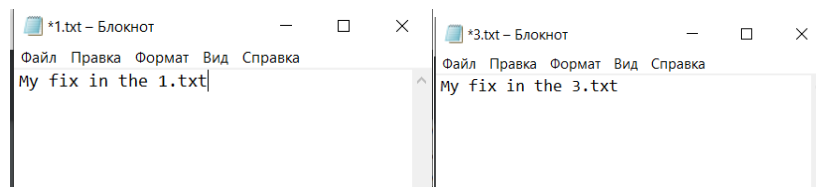


Рисунки 12 и 13. Изменение текстовых файлов

```
Git CMD
* main
C:\2 Семестр\языки прог-ия\лаба 1\1lab2k>git checkout branch_1
Switched to branch 'branch_1'
C:\2 Семестр\языки прог-ия\лаба 1\1lab2k>git add .
C:\2 Семестр\языки прог-ия\лаба 1\1lab2k>git commit -m "Изменения в файлах 1.txt и 3.txt"
[branch_1 98737a9] Изменения в файлах 1.txt и 3.txt
 2 files changed, 2 insertions(+), 1 deletion(-)
C:\2 Семестр\языки прог-ия\лаба 1\1lab2k>
```

Рисунок 14. Коммит изменений

11. Перешла на ветку `branch_2` и также изменила файл `1.txt` на текст “My fix in the 1.txt”, изменила файл `3.txt` на текст “My fix in the 3.txt”, закоммитив изменения:

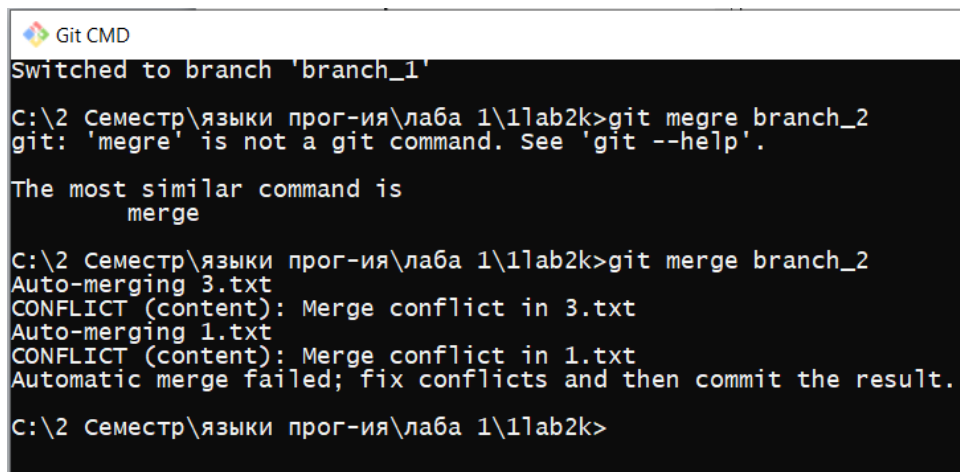


Рисунки 15 и 16. Изменение текстовых файлов

```
Git CMD
Switched to branch 'branch_2'
C:\2 Семестр\языки прог-ия\лаба 1\1lab2k>git add .
C:\2 Семестр\языки прог-ия\лаба 1\1lab2k>git commit -m "Мои изменения в файлах 1.txt и 3.txt"
[branch_2 79f96df] Мои изменения в файлах 1.txt и 3.txt
 2 files changed, 2 insertions(+), 1 deletion(-)
C:\2 Семестр\языки прог-ия\лаба 1\1lab2k>
```

Рисунок 17. Коммит изменений

12. Слила изменения ветки `branch_2` в ветку `branch_1`:



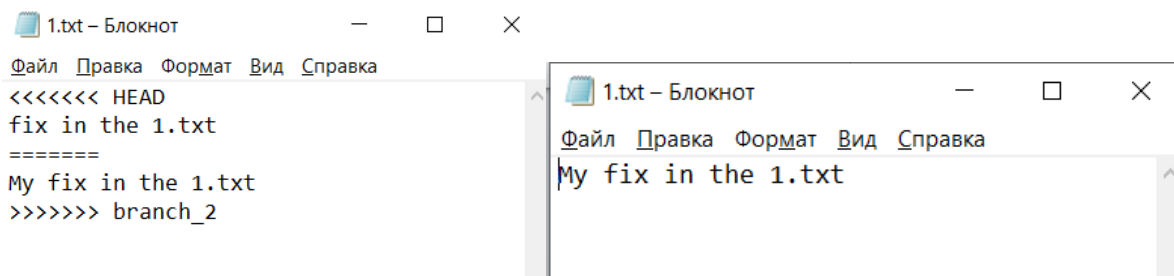
```
Git CMD
Switched to branch 'branch_1'
C:\2 Семестр\языки прог-ия\лаба 1\1lab2k>git megre branch_2
git: 'megre' is not a git command. See 'git --help'.

The most similar command is
    merge

C:\2 Семестр\языки прог-ия\лаба 1\1lab2k>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.
C:\2 Семестр\языки прог-ия\лаба 1\1lab2k>
```

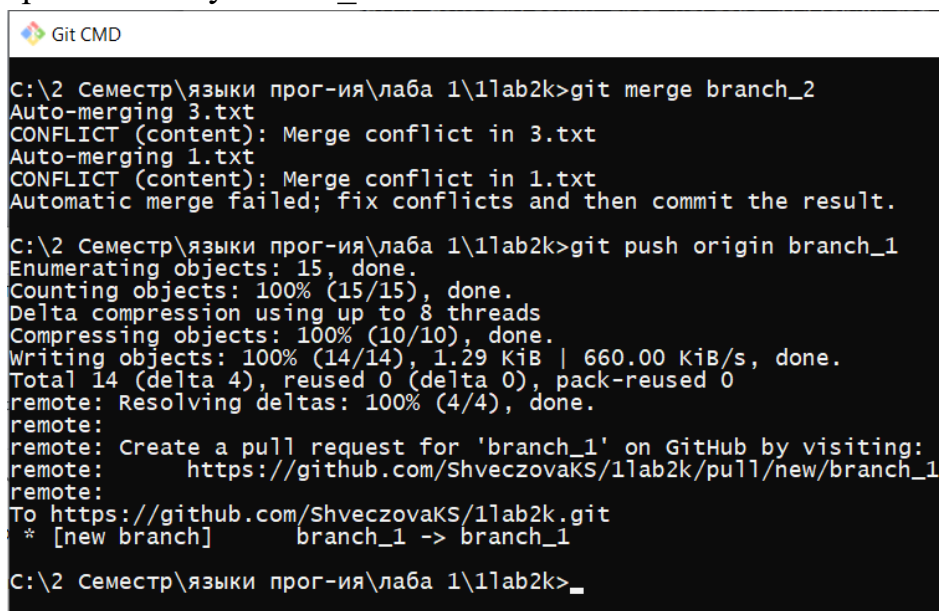
Рисунок 18. Слияние веток

13. Решила конфликт файла 1.txt в ручном режиме:



Рисунки 19 и 20. Решение конфликта файлов

14. Отправила ветку branch_1 на GitHub:



```
Git CMD
C:\2 Семестр\языки прог-ия\лаба 1\1lab2k>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.

C:\2 Семестр\языки прог-ия\лаба 1\1lab2k>git push origin branch_1
Enumerating objects: 15, done.
Counting objects: 100% (15/15), done.
Delta compression using up to 8 threads
Compressing objects: 100% (10/10), done.
Writing objects: 100% (14/14), 1.29 KiB | 660.00 KiB/s, done.
Total 14 (delta 4), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (4/4), done.
remote:
remote: Create a pull request for 'branch_1' on GitHub by visiting:
remote:   https://github.com/ShveczovaKS/1lab2k/pull/new/branch_1
remote:
To https://github.com/ShveczovaKS/1lab2k.git
 * [new branch]      branch_1 -> branch_1
C:\2 Семестр\языки прог-ия\лаба 1\1lab2k>
```

Рисунок 21. Отправление ветки

15. Создала средствами GitHub удаленную ветку branch_3:

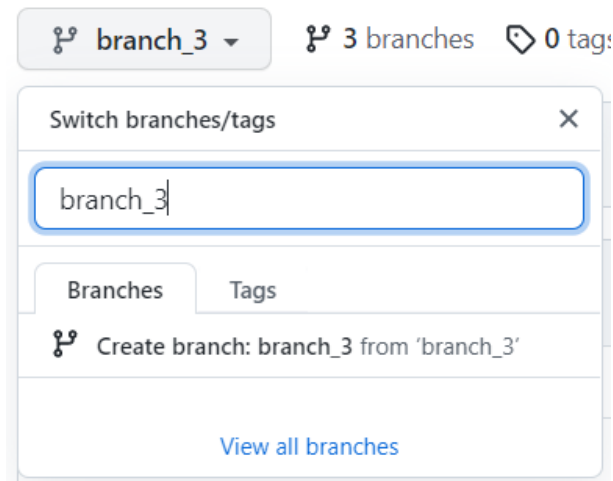


Рисунок 22. Создание удаленной ветки

16. Перешла на ветку branch_3 и добавила файл 2.txt строку "the final fantasy in the 4.txt file":

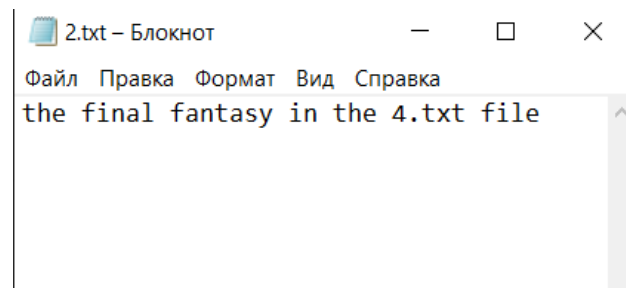


Рисунок 23. Добавление текстового файла

17. Выполнила перемещение ветки master на ветку branch_2 и отправила изменения веток master и branch_2 на GitHub:

```

Git CMD
C:\2 Семестр\языки прог-ия\лаба 1\1lab2k>git add .
C:\2 Семестр\языки прог-ия\лаба 1\1lab2k>git commit -m "Изменения в файле 2.txt"
[branch_3 93ad391] Изменения в файле 2.txt
1 file changed, 1 insertion(+)
C:\2 Семестр\языки прог-ия\лаба 1\1lab2k>git push origin main
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/ShveczovaKS/1lab2k.git
30183e8..d15582d main -> main
C:\2 Семестр\языки прог-ия\лаба 1\1lab2k>git push
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 325 bytes | 325.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/ShveczovaKS/1lab2k.git
98737a9..93ad391 branch_3 -> branch_3

```

Рисунок 24. Коммит изменений

Контрольные вопросы:

1. Что такое ветка?

Ветка в Git — это простой перемещаемый указатель на один из таких коммитов. По умолчанию, имя основной ветки в Git — master.

2. Что такое HEAD?

HEAD — это указатель, задача которого ссылаться на определенный коммит в репозитории. Суть данного указателя можно попытаться объяснить с разных сторон.

Во-первых, HEAD — это указатель на коммит в вашем репозитории, который станет родителем следующего коммита.

Во-вторых, HEAD указывает на коммит, относительно которого будет создана рабочая копия во-время операции checkout . Другими словами, когда вы переключаетесь с ветки на ветку, используя операцию checkout , то в вашем репозитории указатель HEAD будет переключаться между последними коммитами выбираемых вами ветвей.

3. Способы создания веток.

Ветки можно создать с помощью команды “git branch имя_ветки”, и чтобы перейти на неё необходимо использовать команду “git checkout имя_ветки”. Можно же создать ветку и сразу перейти на неё с помощью “git checkout -b имя_ветки”.

4. Как узнать текущую ветку?

С помощью команды git branch высветятся все ветки, текущая будет подкрашена и/или будет со знаком *.

5. Как переключаться между ветками?

Чтобы переходить по веткам, необходимо использовать команду “git checkout имя_ветки”

6. Что такое удаленная ветка?

Это ветка, находящаяся на удалённом сервере GitHub.

7. Что такое ветка отслеживания?

Ветки слежения — это ссылки на определённое состояние удалённых веток. Это локальные ветки, которые напрямую связаны с удалённой веткой. При клонировании репозитория, как правило, автоматически создаётся ветка master, которая следит за origin/master.

8. Как создать ветку отслеживания?

С помощью команды `git checkout -b имя_ветки origin/имя_ветки`.

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

С помощью команды `git push имя_ветки`.

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

`Git fetch` лишь показывает изменения веток на сервере, но не копирует их на локальный репозиторий, в отличие от команды `Git pull`.

11. Как удалить локальную и удаленную ветки?

Можно удалить ветку на удалённом сервере используя параметр `--delete` для команды `git push`.

Локальную ветку можно удалить с помощью команды `git branch -д имя_ветки`.

12. Какие основные типы веток присутствуют в модели `git-flow`? Как организована работа с ветками в модели `git-flow`? В чем недостатки `git-flow`?

`Git Flow` описывает несколько веток для разработки, релизов и взаимодействия между ними. Репозиторий содержит 2 главные ветки:

- `master` (стандартная ветка)
- `develop` (ответвляется от основной, в нее добавляется что-то новое и затем сливается с основной)

Как следствие `master` выступает релизной веткой в этой концепции. В `Git Flow` мы можем использовать такие типы веток как:

- `Feature branches`;
- `Release branches`;
- `Hotfix branches`;

Минусы `git-flow`:

- `git flow` может замедлять работу;
- релизы сложно делать чаще, чем раз в неделю;
- большие функции могут потратить дни на конфликты.

Вывод: были исследованы базовые возможности работы с локальными и удаленными ветками Git.