

Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт цифрового развития
Кафедра инфокоммуникаций

«Лабораторная работа 1.3 Основы ветвления в GitHub»

ОТЧЕТ
по лабораторной работе №3
дисциплины
«Основы программной инженерии»

Выполнил:

Луценко Дмитрий Андреевич
2 курс, группа ПИЖ-б-о-21-1,
09.03.04 «Программная инженерия»,
направленность (профиль) «Разработка
и сопровождение программного
обеспечения», очная форма обучения

(подпись)

Проверил:

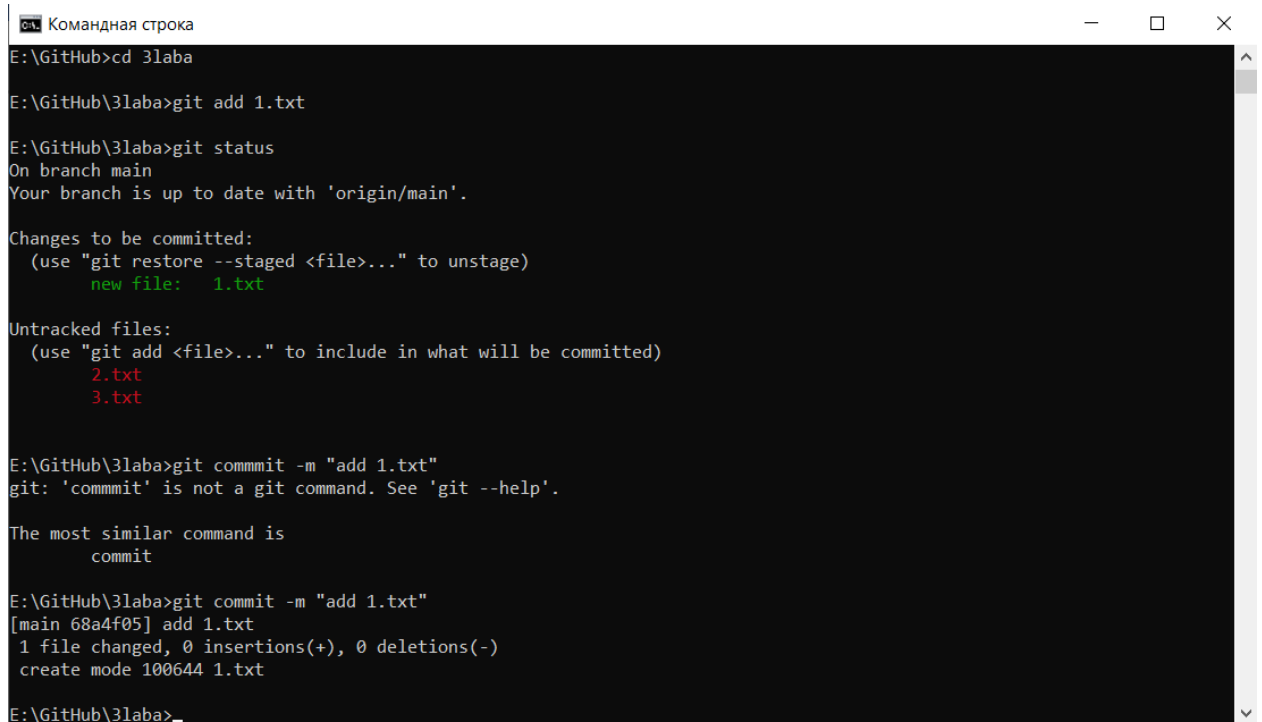
(подпись)

Отчет защищен с оценкой _____ Дата защиты _____

Ставрополь, 2022 г.

Лабораторная работа 1.3 Основы ветвления в GitHub.

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



```
Командная строка
E:\GitHub>cd 3laba

E:\GitHub\3laba>git add 1.txt

E:\GitHub\3laba>git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   1.txt

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        2.txt
        3.txt

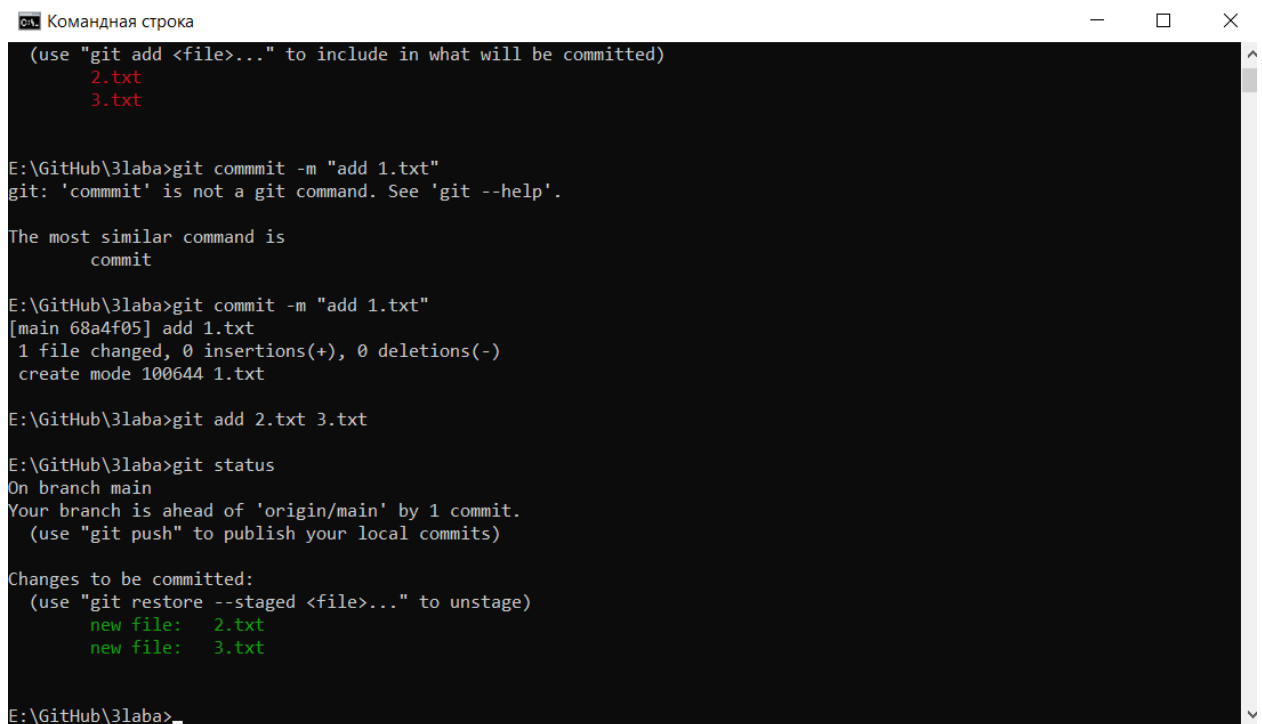
E:\GitHub\3laba>git commit -m "add 1.txt"
git: 'commit' is not a git command. See 'git --help'.

The most similar command is
        commit

E:\GitHub\3laba>git commit -m "add 1.txt"
[main 68a4f05] add 1.txt
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 1.txt

E:\GitHub\3laba>
```

Рисунок 1 – индексация и коммит первого файла



```
(use "git add <file>..." to include in what will be committed)
        2.txt
        3.txt

E:\GitHub\3laba>git commit -m "add 1.txt"
git: 'commit' is not a git command. See 'git --help'.

The most similar command is
        commit

E:\GitHub\3laba>git commit -m "add 1.txt"
[main 68a4f05] add 1.txt
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 1.txt

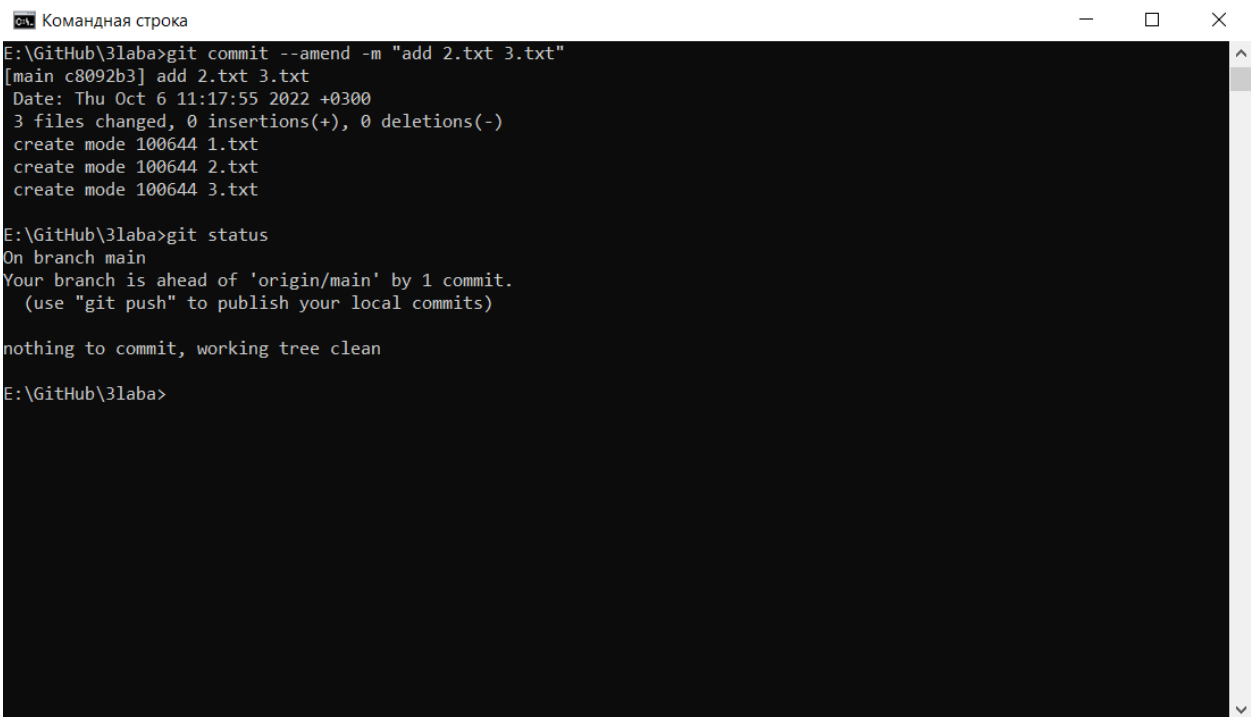
E:\GitHub\3laba>git add 2.txt 3.txt

E:\GitHub\3laba>git status
On branch main
Your branch is ahead of 'origin/main' by 1 commit.
  (use "git push" to publish your local commits)

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   2.txt
        new file:   3.txt

E:\GitHub\3laba>
```

Рисунок 2 – индексация второго и третьего файла



```
E:\GitHub\3laba>git commit --amend -m "add 2.txt 3.txt"
[main c8092b3] add 2.txt 3.txt
Date: Thu Oct 6 11:17:55 2022 +0300
3 files changed, 0 insertions(+), 0 deletions(-)
create mode 100644 1.txt
create mode 100644 2.txt
create mode 100644 3.txt

E:\GitHub\3laba>git status
On branch main
Your branch is ahead of 'origin/main' by 1 commit.
  (use "git push" to publish your local commits)

nothing to commit, working tree clean

E:\GitHub\3laba>
```

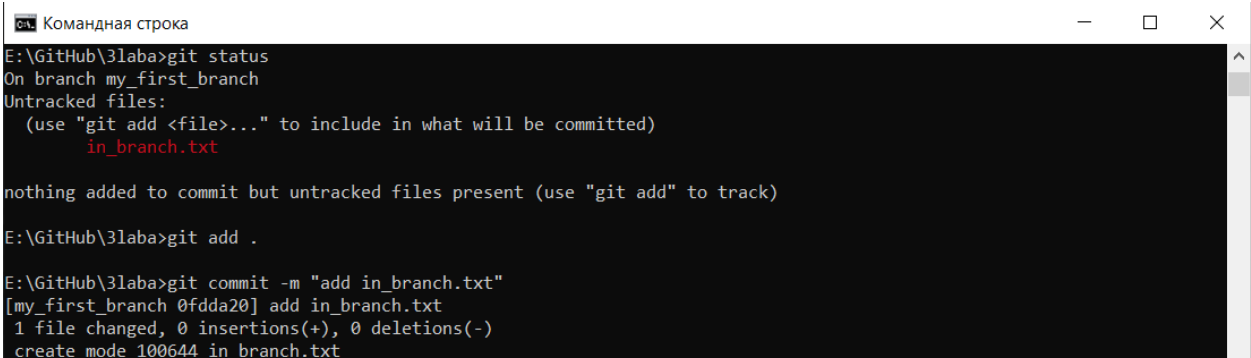
Рисунок 3 – перезапись первого коммита

```
E:\GitHub\3laba>git branch my_first_branch

E:\GitHub\3laba>git checkout my_first_branch
Switched to branch 'my_first_branch'

E:\GitHub\3laba>_
```

Рисунок 4 – создание и переход на новую ветку



```
E:\GitHub\3laba>git status
On branch my_first_branch
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        in_branch.txt

nothing added to commit but untracked files present (use "git add" to track)

E:\GitHub\3laba>git add .

E:\GitHub\3laba>git commit -m "add in_branch.txt"
[my_first_branch 0fdda20] add in_branch.txt
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 in_branch.txt
```

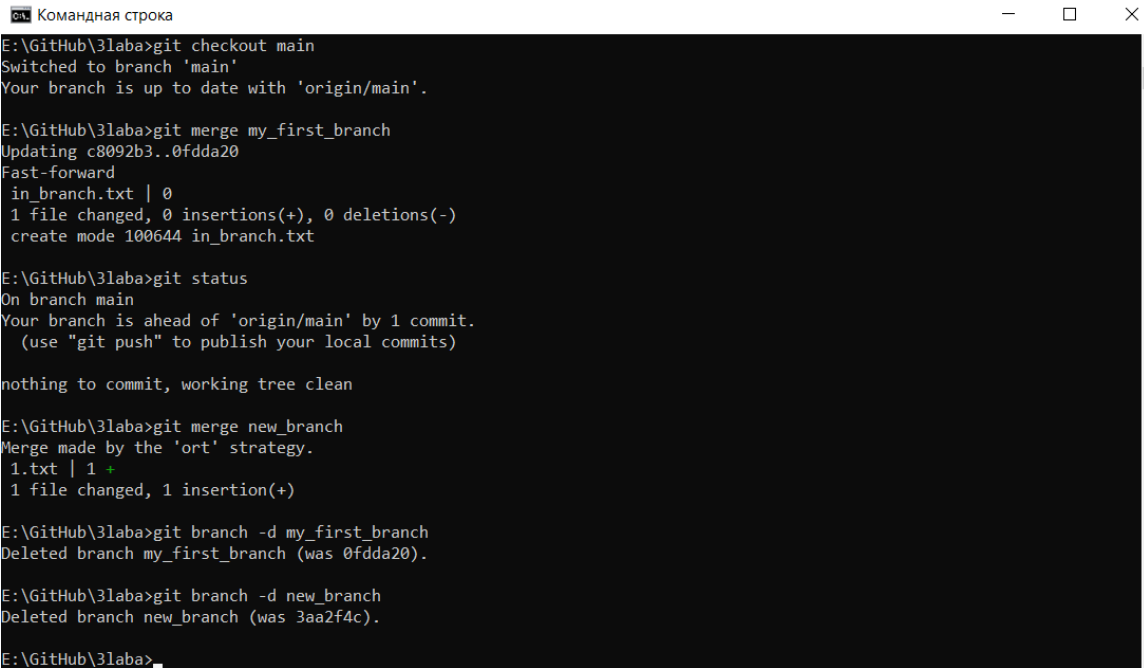
Рисунок 5 – создание коммита на новой ветке и возвращение на ветку main

```
E:\GitHub\3laba>git branch new_branch

E:\GitHub\3laba>git checkout new_branch
Switched to branch 'new_branch'

E:\GitHub\3laba>
```

Рисунок 6 – создание новой ветки new_branch



```
Командная строка
E:\GitHub\3laba>git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

E:\GitHub\3laba>git merge my_first_branch
Updating c8092b3..0fdda20
Fast-forward
 in_branch.txt | 0
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 in_branch.txt

E:\GitHub\3laba>git status
On branch main
Your branch is ahead of 'origin/main' by 1 commit.
  (use "git push" to publish your local commits)

nothing to commit, working tree clean

E:\GitHub\3laba>git merge new_branch
Merge made by the 'ort' strategy.
 1.txt | 1 +
 1 file changed, 1 insertion(+)

E:\GitHub\3laba>git branch -d my_first_branch
Deleted branch my_first_branch (was 0fdda20).

E:\GitHub\3laba>git branch -d new_branch
Deleted branch new_branch (was 3aa2f4c).

E:\GitHub\3laba>
```

Рисунок 7 – слияние и удаление веток

```
Командная строка
Deleted branch new_branch (was 3aa2f4c).

E:\GitHub\3laba>git branch branch_1

E:\GitHub\3laba>git branch branch_2

E:\GitHub\3laba>git checkout branch_1
Switched to branch 'branch_1'

E:\GitHub\3laba>git add .

E:\GitHub\3laba>git commit -m "changed 1.txt and 3.txt"
[branch_1 9d60f6e] changed 1.txt and 3.txt
 2 files changed, 2 insertions(+), 1 deletion(-)

E:\GitHub\3laba>git push
fatal: The current branch branch_1 has no upstream branch.
To push the current branch and set the remote as upstream, use

    git push --set-upstream origin branch_1

To have this happen automatically for branches without a tracking
upstream, see 'push.autoSetupRemote' in 'git help config'.

E:\GitHub\3laba>git push --set-upstream origin branch_1
Enumerating objects: 12, done.
Counting objects: 100% (10/10), done.
Delta compression using up to 8 threads
Compressing objects: 100% (4/4), done.
```

Рисунок 8 – создание новых веток и коммит на ветке branch_1

```
E:\GitHub\3laba>git merge branch_2
Auto-merging 1.txt
CONFLICT (content): Merge conflict in 1.txt
Auto-merging 3.txt
CONFLICT (content): Merge conflict in 3.txt
Automatic merge failed; fix conflicts and then commit the result.
```

Рисунок 9 – конфликт файлов при слиянии веток

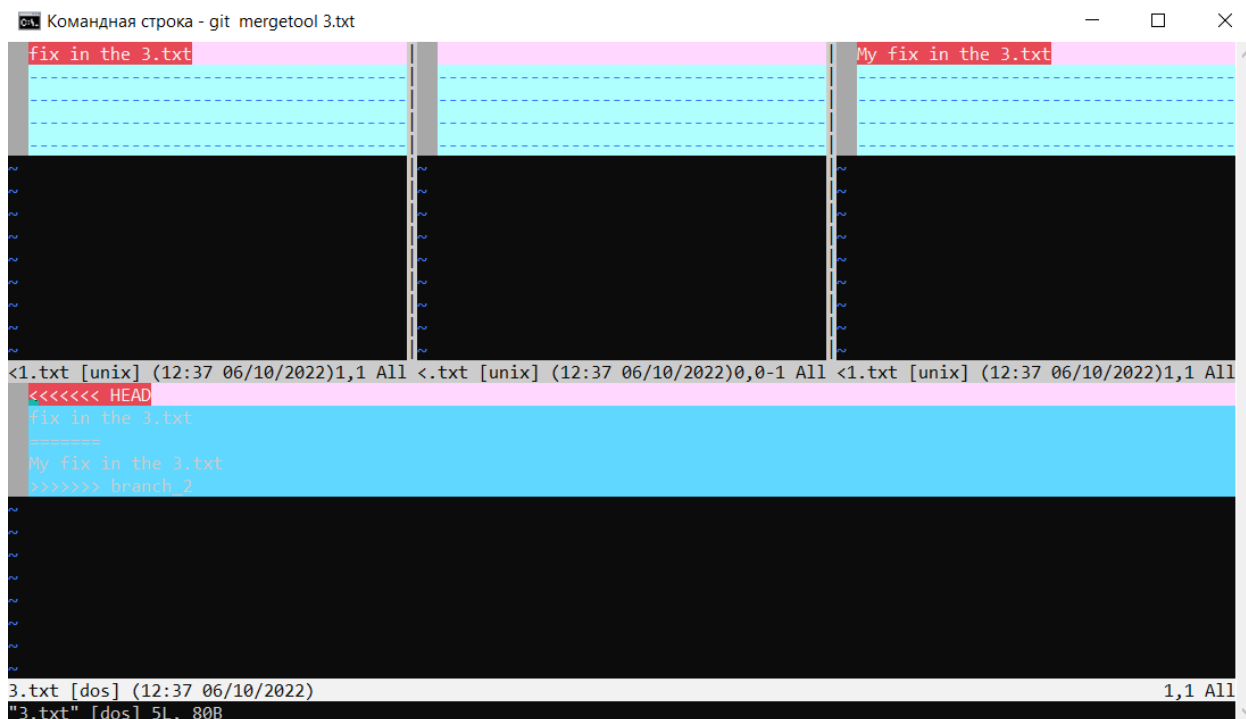


Рисунок 10 – утилита Meld для решения конфликтов файлов

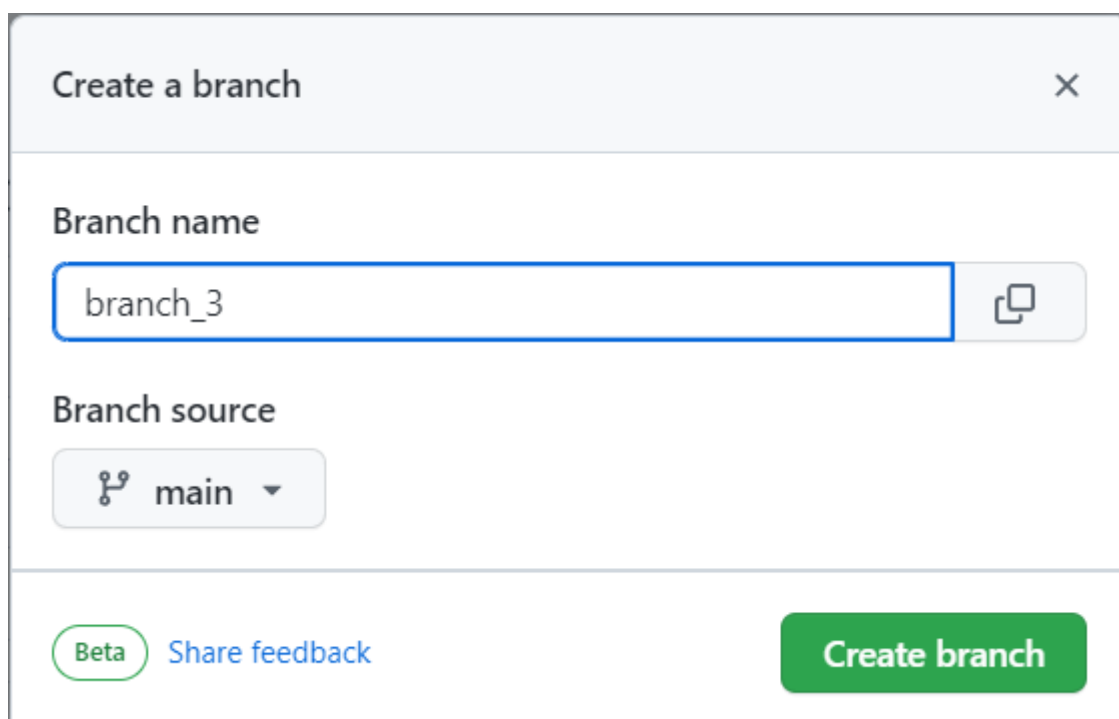


Рисунок 11 – создание ветки средствами GitHub

```
E:\GitHub\3laba>git pull
From https://github.com/DFooRS/3laba
 * [new branch]      branch_3    -> origin/branch_3
Already up to date.

E:\GitHub\3laba>git checkout branch_3
Switched to a new branch 'branch_3'
branch 'branch_3' set up to track 'origin/branch_3'.
```

Рисунок 12 – переход на ветку branch_3

```
E:\GitHub\3laba>git checkout main
Switched to branch 'main'
Your branch is ahead of 'origin/main' by 3 commits.
  (use "git push" to publish your local commits)

E:\GitHub\3laba>git rebase branch_2
Successfully rebased and updated refs/heads/main.

E:\GitHub\3laba>
```

Рисунок 13 – перебазирование ветки

Ответы на вопросы:

1. Ветка – это простой перемещаемый указатель на коммит
2. HEAD – это указатель, задача которого ссылаться на определенный коммит
3. Создать ветку можно через консоль командой `git branch <name>`, либо через интерфейс GitHub
4. Чтобы узнать текущую ветку, необходимо ввести `git status`
5. Для переключения между ветками необходимо ввести команду `git checkout <name>`
6. Удалённые ветки – это ветки в удалённых репозиториях
7. Ветки слежения – это локальные ветки, напрямую связанные с удалёнными
8. Для создания ветки отслеживания нужно ввести `git checkout –track origin/name`, или ввести сокращение `git checkout name`

9. Для отправки изменений из локальной ветки нужно ввести `git push <remote> <branch>`
10. `Git pull` берет все новые коммиты и сливает их в одну ветку. `Git fetch` же только берет изменения с сервера и сохраняет их в локальном репозитории
11. Для удаления удаленной ветки нужно ввести `git push origin -delete <name>`, для удаления локальной ветки нужно ввести `git branch -d <name>`
12. Основные типы веток `git-flow`: `develop`, `release`, `feature`, `hotfix`. Работа с ветками организована следующим образом: из ветки `main` формируется ветка `develop`, из неё создается ветка `release` и `feature`. Когда работа над `feature` завершается, она сливается в ветку `develop`. Когда работа над веткой `release` завершается, она сливается с ветками `develop` и `main`. Если в `main` обнаруживается проблема, из неё создается `hotfix`. Когда работа с `hotfix` завершается, она сливается с ветками `develop` и `main`. Недостатки `git-flow`: сложно делать часто релизы, большие функции могут потратить много времени на мёрж, история имеет кучу мёржей и затрудняет просмотр истории проектов
13. В `GitKraken` есть инструменты для слияния веток, перебазирования, отслеживания