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

Институт цифрового развития
Кафедра инфокоммуникаций

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ №1.2
дисциплины «Основы кроссплатформенного программирования»

Выполнил:
Медяник Даниил Владимирович
1 курс, группа ИТС-б-о-22-1,
11.03.02 «Инфокоммуникационные технологии и системы связи»,
направленность (профиль)
«Инфокоммуникационные системы и сети», очная форма обучения

_____
(подпись)

Руководитель практики:
Воронкин Р.А., канд. тех. наук, доцент,
доцент кафедры инфокоммуникаций

_____
(подпись)

Отчет защищен с оценкой _____ Дата защиты_____

Ставрополь, 2023 г.

Тема: исследование возможностей Git для работы с локальными репозиториями.

Цель работы: исследовать базовые возможности системы контроля версий Git для работы с локальными репозиториями.

Порядок выполнения работы:

Задание 1.

Создал новый репозиторий и клонировал его на свой компьютер.

```
C:\Users\Gaming-PC>git clone https://github.com/DaniilMedy/laba-1.2.git
Cloning into 'laba-1.2'...
remote: Enumerating objects: 8, done.
remote: Counting objects: 100% (8/8), done.
remote: Compressing objects: 100% (7/7), done.
remote: Total 8 (delta 1), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (8/8), done.
Resolving deltas: 100% (1/1), done.
```

Рисунок 1. Новый репозиторий

Задание 2. Проработать примеры лабораторной работы.

Пример 1. Просмотр истории коммитов

```
C:\Users\Gaming-PC\laba-1.2>git log
commit 3509e4074e1d998b8d4b90eb4308d65a58b37464 (HEAD -> main, origin/main, origin/HEAD)
Author: DaniilMedy <135615674+DaniilMedy@users.noreply.github.com>
Date:   Mon Jun 5 17:08:06 2023 +0300

    Add

commit b3a10fabb7af4d7f4320937ebb6ec35fc4bb760b
Author: DaniilMedy <135615674+DaniilMedy@users.noreply.github.com>
Date:   Mon Jun 5 17:04:06 2023 +0300

    Initial commit
```

Рисунок 2. Команда “git log”

Вывод только двух записей. Команда - git log -p -2.

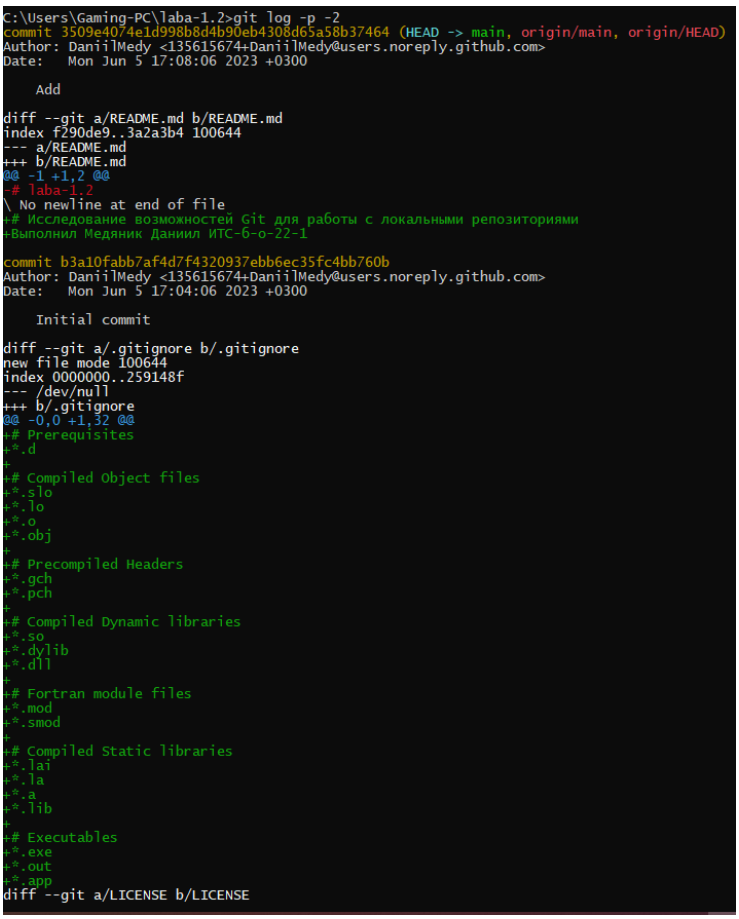

Рисунок 3. Команда “git log -p -2”

Чтобы увидеть сокращенную статистику для каждого коммита, можно использовать опцию --stat :

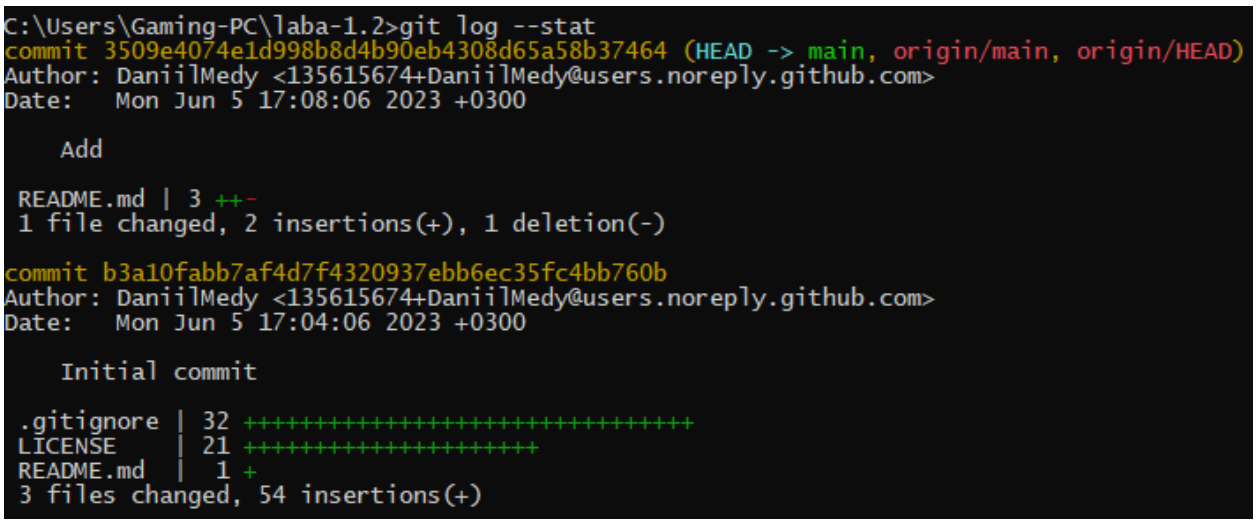

Рисунок 4. Команда “git log --stat”

Опция oneline выводит каждый коммит в одну строку, что может быть очень удобным если вы просматриваете большое количество коммитов.

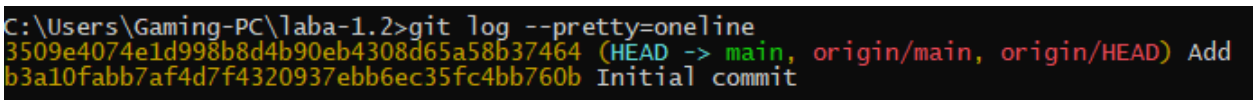

Рисунок 5. Команда “git log –pretty=oneline”

Наиболее интересной опцией является format , которая позволяет указать формат для вывода информации.

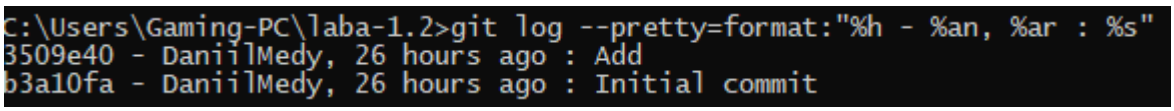

Рисунок 6. Команда “git log --pretty=format:"%h - %an, %ar : %s"”

Опции oneline и format являются особенно полезными с опцией --graph команды log .

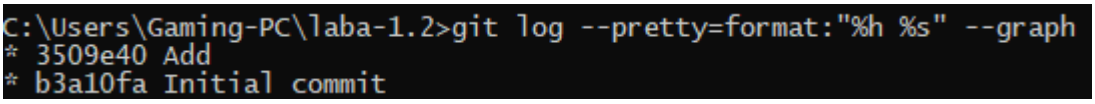

Рисунок 7. Команда “git log --pretty=format:"%h %s" --graph”

Ограничение вывода

Опции для ограничения вывода по времени, такие как --since и --until, являются очень удобными. Например, следующая команда покажет список коммитов, сделанных за последние две недели:

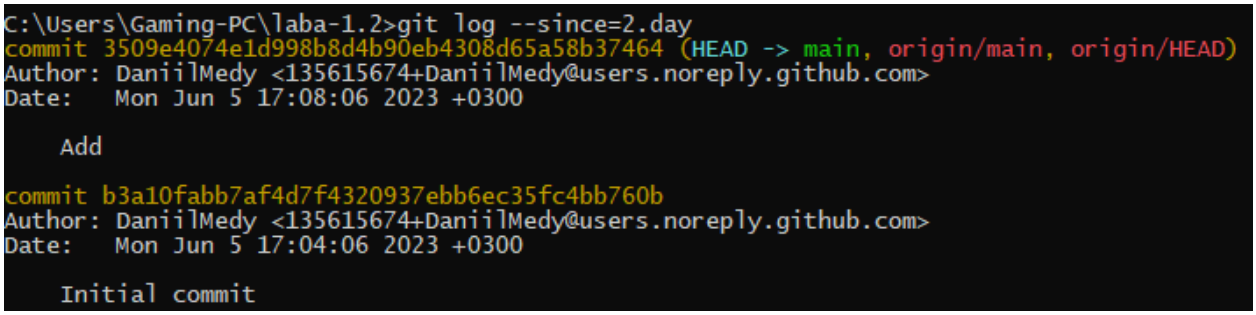

Рисунок 8. Команда “git log --since=1.day”

Операции отмены

Если вы хотите переделать коммит – то внесите необходимые изменения, добавьте их в индекс и сделайте коммит ещё раз, указав параметр --amend:

```
C:\Users\Gaming-PC\laba-1.2>git commit -m "test --amend"
[main 53e0e86] test --amend
 1 file changed, 1 insertion(+), 1 deletion(-)

C:\Users\Gaming-PC\laba-1.2>git commit -m "test --amend" --amend
[main c464678] test --amend
 Date: Tue Jun 6 19:17:09 2023 +0300
 1 file changed, 1 insertion(+), 1 deletion(-)
```

Рисунок 9. Команда “git --amend”

Просмотр удалённых репозиториев

Для того, чтобы просмотреть список настроенных удалённых репозиториев, вы можете запустить команду git remote .

```
C:\Users\Gaming-PC\laba-1.2>git remote  -v
origin  https://github.com/DaniilMedy/laba-1.2.git (fetch)
origin  https://github.com/DaniilMedy/laba-1.2.git (push)
```

Рисунок 10. Команда “git remote -v”

Просмотр удаленного репозитория

Если хотите получить побольше информации об одном из удалённых репозиториев, вы можете использовать команду git remote show <remote>.

```
C:\Users\Gaming-PC\laba-1.2>git remote show origin
* remote origin
  Fetch URL: https://github.com/DaniilMedy/laba-1.2.git
  Push  URL: https://github.com/DaniilMedy/laba-1.2.git
  HEAD branch: main
  Remote branch:
    main tracked
  Local branch configured for 'git pull':
    main merges with remote main
  Local ref configured for 'git push':
    main pushes to main (fast-forwardable)
```

Рисунок 11. Команда “git show origin”

Работа с тегами

Просмотр списка тегов

Просмотреть список имеющихся тегов в Git можно очень просто. Достаточно набрать команду git tag (параметры -l и --list опциональны):

```
C:\Users\Gaming-PC\laba-1.2>git tag
v1.1
v1.2
v1.3
v1.4
v1.5
```

Рисунок 12. Команда “git tag”

Создание аннотированного тега в Git выполняется легко. Самый простой способ — это указать -a при выполнении команды tag

```
C:\Users\Gaming-PC\laba-1.2>git tag -a v1.5 -m "klava vvod"
```

Рисунок 13. Создание тега.

По умолчанию, команда git push не отправляет теги на удалённые сервера. После создания теги нужно отправлять явно на удалённый сервер. Процесс аналогичен отправке веток - достаточно выполнить команду git push origin <tagname>

```
C:\Users\Gaming-PC\laba-1.2>git push origin --tags
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 12 threads
Compressing objects: 100% (5/5), done.
Writing objects: 100% (5/5), 666 bytes | 666.00 KiB/s, done.
Total 5 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/DaniilMedy/laba-1.2.git
 * [new tag]         v1.1 -> v1.1
 * [new tag]         v1.2 -> v1.2
 * [new tag]         v1.3 -> v1.3
 * [new tag]         v1.4 -> v1.4
 * [new tag]         v1.5 -> v1.5
```

Рисунок 14**.** Команда “git push origin –tags”

Задание 3.

Написал небольшую программу в новом файле programm.cpp, сделал не менее 7-ми коммитов с 3-мя тегами. Использование команды “git log –graph –pretty=oneline –abbrev-commit”.

```
C:\Users\Gaming-PC\laba-1.2>git log --graph --pretty=oneline --abbrev-commit
* 7ce4498 (HEAD -> main) scan_f
* 081f719 (tag: v1.5) printf
* 8c3c4cb (tag: v1.4) for
* 3499d70 (tag: v1.3) peremennie
* 8b88275 (tag: v1.2) const
* 8ae3f33 main
* d9b5455 (tag: v1.1) include
```

Рисунок 15. История хранилища.

Задание 5.

Посмотрел содержимое коммитов командой git show <ref>, где <ref>:

1) HEAD : последний коммит;

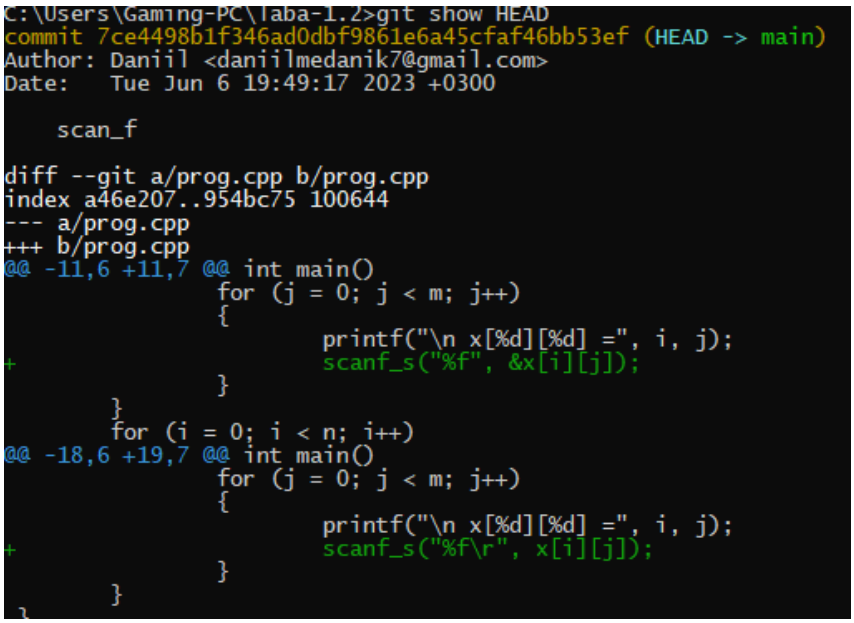

Рисунок 16. Последний коммит.

2) HEAD~1 : предпоследний коммит (и т. д.);

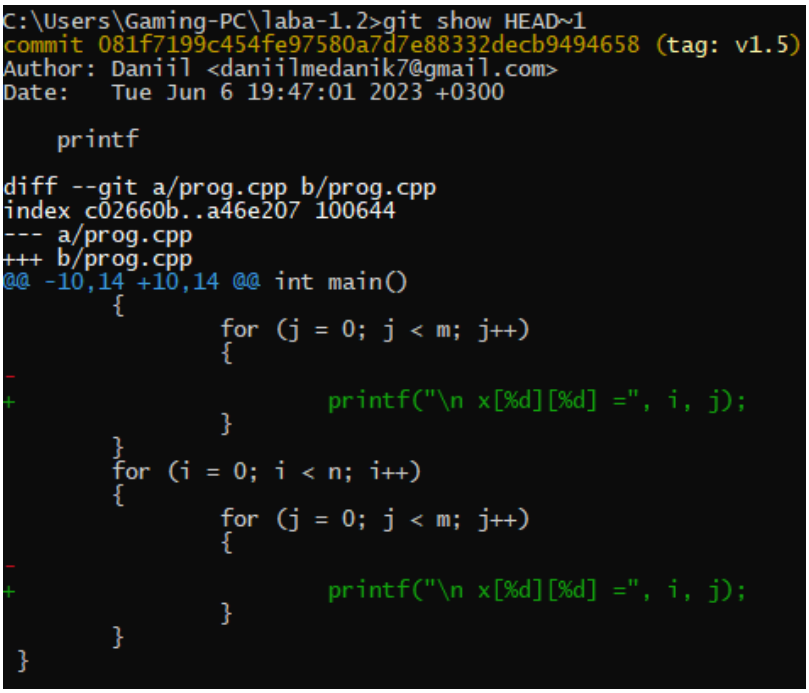

Рисунок 17. Предпоследний коммит.

3) 081f719: коммит с указанным хэшем.

```
C:\Users\Gaming-PC\laba-1.2>git show 081f719
commit 081f7199c454fe97580a7d7e88332decb9494658 (tag: v1.5)
Author: Daniil <daniilmedanik7@gmail.com>
Date:   Tue Jun 6 19:47:01 2023 +0300

    printf

diff --git a/prog.cpp b/prog.cpp
index c02660b..a46e207 100644
--- a/prog.cpp
+++ b/prog.cpp
@@ -10,14 +10,14 @@ int main()
        {
                for (j = 0; j < m; j++)
                {
-
+                       printf("\n x[%d][%d] =", i, j);
                }
        }
        for (i = 0; i < n; i++)
        {
                for (j = 0; j < m; j++)
                {
-
+                       printf("\n x[%d][%d] =", i, j);
                }
        }
 }
```

Рисунок 18. Коммит с указанным хэшем.

Задание 6. Откат к заданной версии.

1.1. Удалил весь программный код с файла main.cpp и сохранил его.

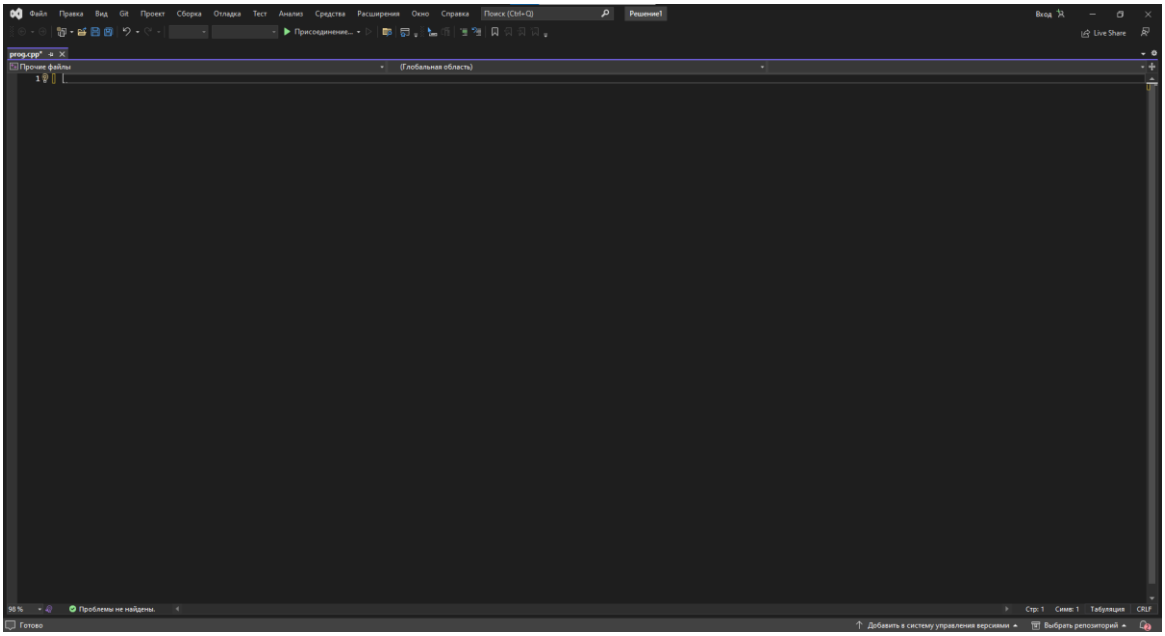

Рисунок 19. Удаление программ

1.2. Удалил это изменение с помощью команды *git checkout -- main.cpp.docx.*

```
C:\Users\Gaming-PC\laba-1.2>git checkout -- prog.cpp

C:\Users\Gaming-PC\laba-1.2>
```

Рисунок 20. Команда “git checkout – programm.cpp”

Код вернулся к прежнему состоянию.

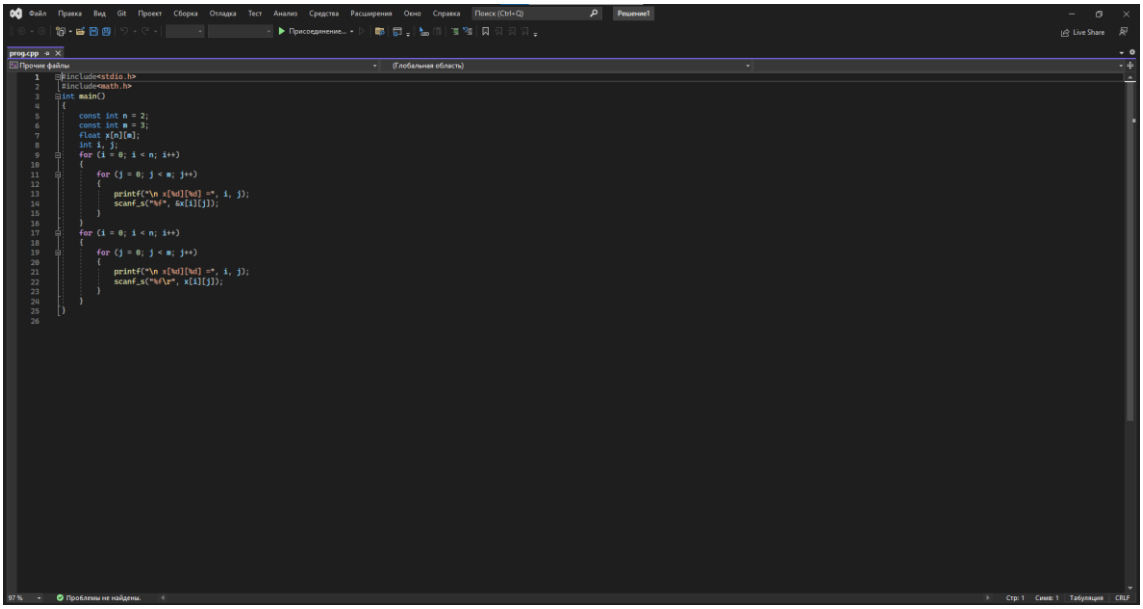

Рисунок 21. Восстановление программы.

1.3. Вновь повторил пункт 1.1. и сделал коммит.

```
C:\Users\Gaming-PC\laba-1.2>git add prog.cpp

C:\Users\Gaming-PC\laba-1.2>git commit -m "delete"
[main e3de155] delete
 1 file changed, 25 deletions(-)
```

Рисунок 22. Коммит.

1.4. Откатить состояние хранилища к предыдущей версии командой: *git reset --hard HEAD~1* .

```
C:\Users\Gaming-PC\laba-1.2>git reset --hard HEAD~1
HEAD is now at 7ce4498 scan_f
```

Рисунок 23. Возвращение к предпоследней версии коммита.

Ссылка на репозиторий: https://github.com/DaniilMedy/laba-1.2

Ответы на контрольные вопросы:

1) Как выполнить историю коммитов в Git? Какие существуют дополнительные опции для просмотра истории коммитов?

Историю коммитов можно выполнить с помощью команды git log.

Дополнительные опции для просмотра истории:

%H, %h, %T, %t, %P, %p тд.

-p, --stat, --shortstat, --name-only, --name-status и тд.

2) Как ограничить вывод при просмотре истории коммитов?

Ограничить вывод при просмотре истории коммитов можно с помощью команды *git log –n*, где n – число последних коммитов.

3) Как внести изменения в уже сделанный коммит?

Если вы хотите переделать коммит — внесите необходимые изменения, добавьте их в индекс и сделайте коммит ещё раз, указав

параметр *--amend* : *git commit –amend.*

4) Как отменить индексацию файла в Git?

Отменить индексацию файла можно с помощью команды: *git reset HEAD <file>*.

5) Как отменить изменения в файле?

Отменить изменения в файле можно с помощью команды: *git checkout -- <file>*

6) Что такое удаленный репозиторий Git?

Удалённые репозитории представляют собой версии вашего

проекта, сохранённые в интернете или ещё где-то в сети.

7) Как выполнить просмотр удаленных репозиториев данного локального репозитория?

Выполнить просмотр удаленных репозиториев данного локального репозитория можно с помощью команды: *git remote.*

8) Как добавить удаленный репозиторий для данного локального репозитория?

Для того, чтобы добавить удалённый репозиторий и присвоить ему имя (shortname), просто выполните команду *git remote add <shortname> <url>*.

9) Как выполнить отправку/получение изменений с удаленного репозитория?

Для получения данных из удалённых проектов, следует выполнить: *git fetch [remote-name]*.

Для отправки изменений в удаленный репозиторий используется команда: *git push <remote-name> <branch-name>*

10) Как выполнить просмотр удаленного репозитория?

Если хотите получить побольше информации об одном из удалённых репозиториев, вы можете использовать команду: *git remote show <remote>*.

11) Каково назначение тэгов Git?

Git имеет возможность помечать определённые моменты в истории как важные. Для таких случаев были придуманы тэги.

12) Как осуществляется работа с тэгами Git?

Просмотреть список имеющихся тегов в Git можно очень просто. Достаточно набрать команду *git tag*.

Создание аннотированного тега в Git выполняется легко. Самый простой способ — это указать *-a* при выполнении команды *tag.*

С помощью команды *git show* вы можете посмотреть данные тега вместе с коммитом.

По умолчанию, команда *git push* не отправляет теги на удалённые сервера. После создания теги нужно отправлять явно на удалённый сервер. Процесс аналогичен отправке веток — достаточно выполнить команду *git push origin <tagname>*.

Для удаления тега в локальном репозитории достаточно выполнить команду *git tag –d <tagname>* .

Если вы хотите получить версии файлов, на которые указывает тег, то вы можете сделать *git checkout <tagname>* для тега.

13) Самостоятельно изучите назначение флага --prune в командах git fetch и git push. Каково назначение этого флага?

Git prune-это команда, которая удаляет все файлы, недоступные из текущей ветви. Команда prune полезна, когда в вашем рабочем каталоге много файлов, которые вы не хотите хранить.

git fetch --prune делает то же самое: удалет ссылки на ветки, которые не существуют на удаленном компьютере.

Опция *–prune* в команде *git push* удалит ветку из удаленного репозитория, если в локальном репозитории не существует ветки с таким именем.

Вывод: исследовал базовые возможности системы контроля версий Git для работы с локальными репозиториями.