

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙ-  
СКОЙ ФЕДЕРАЦИИ**

**Федеральное государственное автономное  
образовательное учреждение высшего образования  
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»**

**Кафедра инфокоммуникаций**

**Основы кроссплатформенного программирования**

**Отчет по лабораторной работе №1.2**

Исследование  
возможностей Git для работы с  
локальными репозиториями

Выполнил студент группы

ИВТ-б-о-21-1

Богадуров В.И. « » \_\_\_\_\_ 20\_\_ г.

Подпись студента \_\_\_\_\_

Работа защищена « » \_\_\_\_\_ 20\_\_ г.

Проверил доцент

Кафедры инфокоммуникаций, старший  
преподаватель

Воронкин Р.А.

\_\_\_\_\_  
(подпись)

Ставрополь 2022

**Тема:** Исследование возможностей Git для работы с локальными репозиториями.

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

1) Необходимо создать новый репозиторий, в файле README.md указать информацию о обучающемся и скопировать репозиторий себе на компьютер.

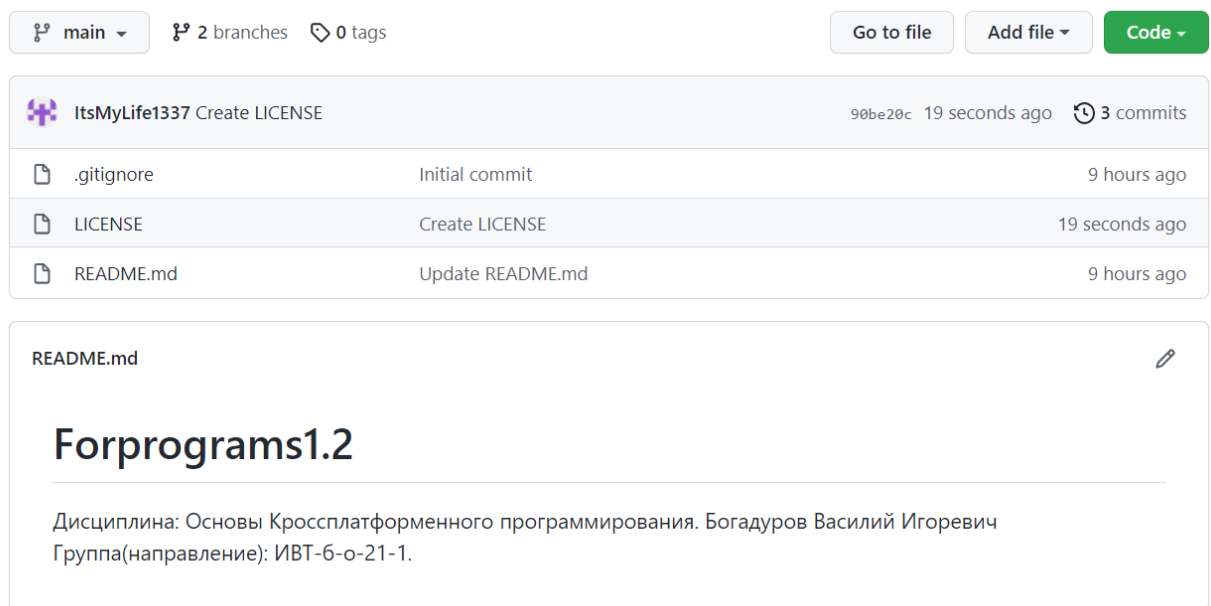


Рисунок 1 – Новый репозиторий

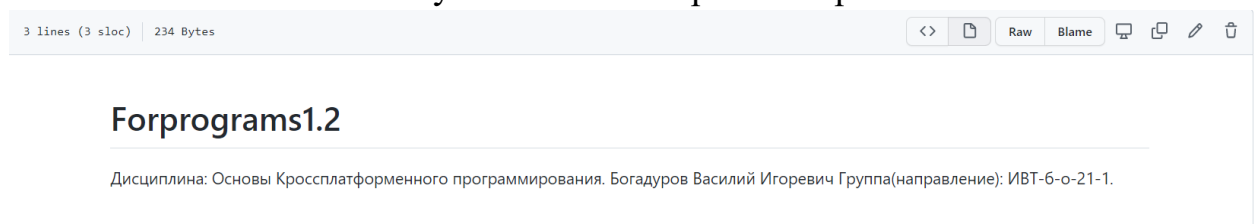


Рисунок 2 – Изменённый файл README

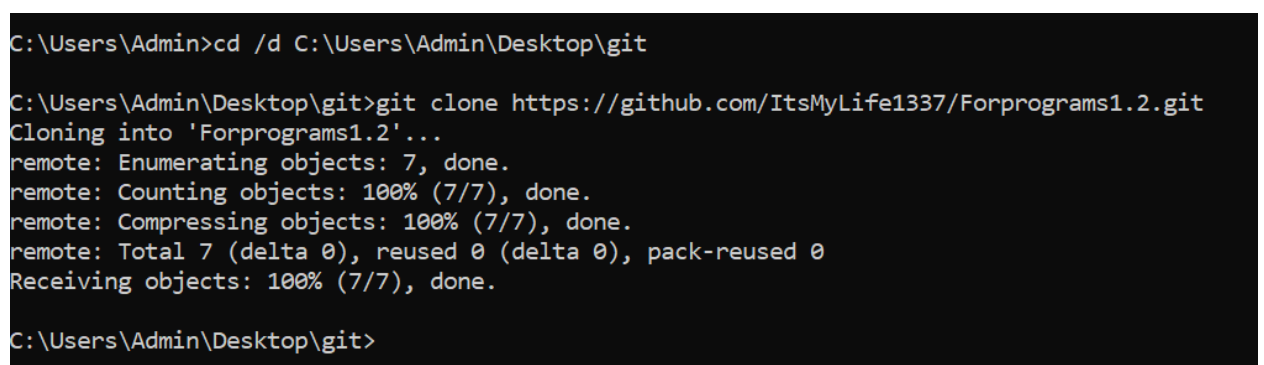


Рисунок 3 – Копирование репозитория на компьютер

## 2) Проработка примеров в лабораторной работе.

### 1. Использование команды git log:

```
C:\Users\Admin\Desktop\git\Forprograms1.2>git log
commit 4090fbb7e26e11377995bf83542bbd86ca73ca8b (HEAD -> main, origin/main, origin/HEAD)
Author: ItsMyLife1337 <99473212+ItsMyLife1337@users.noreply.github.com>
Date:   Fri Mar 11 11:59:36 2022 +0300

    Update README.md

commit 411cfba8b77214732d0b32131ae9d6aa1db25ceb
Author: ItsMyLife1337 <99473212+ItsMyLife1337@users.noreply.github.com>
Date:   Fri Mar 11 11:46:53 2022 +0300

    Initial commit

C:\Users\Admin\Desktop\git\Forprograms1.2>
```

Рисунок 4 – Вывод команды git log в консоли

### 2. Использование команды git log -p -2(где 2 количество выведенных записей):

---

cmd. Командная строка - git log -p -2

```
commit 4090fbb7e26e11377995bf83542bbd86ca73ca8b (HEAD -> main, origin/main, origin/HEAD)
Author: ItsMyLife1337 <99473212+ItsMyLife1337@users.noreply.github.com>
Date:   Fri Mar 11 11:59:36 2022 +0300

    Update README.md

diff --git a/README.md b/README.md
index 8a87455..54cba5c 100644
--- a/README.md
+++ b/README.md
@@ -1,2 +1,3 @@
 # Forprograms1.2
-Для лабораторной работы 1.2
+Дисциплина: Основы Кроссплатформенного программирования.
+Богадунов Василий Игоревич Группа(направление): ИБТ-6-о-21-1.

commit 411cfba8b77214732d0b32131ae9d6aa1db25ceb
Author: ItsMyLife1337 <99473212+ItsMyLife1337@users.noreply.github.com>
Date:   Fri Mar 11 11:46:53 2022 +0300

    Initial commit

diff --git a/.gitignore b/.gitignore
new file mode 100644
index 0000000..259148f
--- /dev/null
+++ b/.gitignore
@@ -0,0 +1,32 @@
+# Prerequisites
.
```

Рисунок 5 – Вывод команды git log -p -2 в консоль

### 3. Использование команды git log --stat:

```
C:\Users\Admin\Desktop\git\Forprograms1.2>git log --stat
commit 4090fbb7e26e11377995bf83542bbd86ca73ca8b (HEAD -> main, origin/main, origin/HEAD)
Author: ItsMyLife1337 <99473212+ItsMyLife1337@users.noreply.github.com>
Date:   Fri Mar 11 11:59:36 2022 +0300

    Update README.md

 README.md | 3 ++-
 1 file changed, 2 insertions(+), 1 deletion(-)

commit 411cfba8b77214732d0b32131ae9d6aa1db25ceb
Author: ItsMyLife1337 <99473212+ItsMyLife1337@users.noreply.github.com>
Date:   Fri Mar 11 11:46:53 2022 +0300

    Initial commit

.gitignore | 32 ++++++
 README.md |  2 ++
 2 files changed, 34 insertions(+)
```

Рисунок 6 – Вывод команды git log --stat

### 4. Использование команды git log --pretty=format:"%h - %an, %ar : %s"

```
C:\Users\Admin\Desktop\git\Forprograms1.2>git log --pretty=format:"%h - %an, %ar : %s"
4090fbb - ItsMyLife1337, 9 hours ago : Update README.md
411cfba - ItsMyLife1337, 10 hours ago : Initial commit
```

Рисунок 7 – Вывод команды git log --pretty в консоль

### 5. Использование команды git log -s

```
C:\Users\Admin\Desktop\git\Forprograms1.2>git log -s
commit 4090fbb7e26e11377995bf83542bbd86ca73ca8b (HEAD -> main, origin/main, origin/HEAD)
Author: ItsMyLife1337 <99473212+ItsMyLife1337@users.noreply.github.com>
Date:   Fri Mar 11 11:59:36 2022 +0300

    Update README.md

commit 411cfba8b77214732d0b32131ae9d6aa1db25ceb
Author: ItsMyLife1337 <99473212+ItsMyLife1337@users.noreply.github.com>
Date:   Fri Mar 11 11:46:53 2022 +0300

    Initial commit
```

Рисунок 8 – Вывод команды git log -s в консоль

### 6. Использование команды git remote -v

```
C:\Users\Admin\Desktop\git\Forprograms1.2>git remote -v
origin https://github.com/ItsMyLife1337/Forprograms1.2.git (fetch)
origin https://github.com/ItsMyLife1337/Forprograms1.2.git (push)
```

Рисунок 9 – Вывод команды git remote -v в консоль

## 7. Использование команды git remote show origin

```
C:\Users\Admin\Desktop\git\Forprograms1.2>git remote show origin
remote origin
Fetch URL: https://github.com/ItsMyLife1337/Forprograms1.2.git
Push URL: https://github.com/ItsMyLife1337/Forprograms1.2.git
HEAD branch: main
Remote branches:
  add-license-1 new (next fetch will store in remotes/origin)
  main          tracked
Local branch configured for 'git pull':
  main merges with remote main
Local ref configured for 'git push':
  main pushes to main (local out of date)
```

Рисунок 10 – Вывод команды git remote show origin

2) Написать небольшую программу на выбранном языке программирования(C++). Фиксировать изменения при написании программы в локальном репозитории. Должно быть сделано не менее 7 коммитов, отмеченных не менее 3 тэгами.

1-2. Создал проект в VisualStudio и написал исходную программу.

git > Forprograms1.2 > test1				Поиск: test1
Имя	Дата изменения	Тип		
x64	11.03.2022 21:50	Папка с файлами		
test1.cpp	11.03.2022 21:50	C++ Source		
test1.sln	11.03.2022 21:49	Visual Studio Solution		
test1.vcxproj	11.03.2022 21:49	VC++ Project		
test1.vcxproj.filters	11.03.2022 21:49	VC++ Project Filters F...		
test1.vcxproj.user	11.03.2022 21:49	Per-User Project Opti...		

Рисунок 11 – Созданный проект

```
test1.cpp  -p  X
test1
1  #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6      cout << "Hello World!\n";
7  }
8
```

Рисунок 12 – Написанная программа

3. Создал первый тег.

```
C:\Users\Admin\Desktop\git\Forprograms1.2>git tag -a v1.1 -m "v1.1"

C:\Users\Admin\Desktop\git\Forprograms1.2>git tag
v1.1

C:\Users\Admin\Desktop\git\Forprograms1.2>
```

Рисунок 13 – Созданный тег

```
C:\Users\Admin\Desktop\git\Forprograms1.2>git tag
v1.1

C:\Users\Admin\Desktop\git\Forprograms1.2>git show
commit 53e15c315318237bc822e86376d9336d4751b48d (HEAD -> main)
Author: ItsMyLife1337 <itsmylife1337@yandex.ru>
Date:   Fri Mar 11 22:06:29 2022 +0300

    test1

diff --git a/test1/test1.cpp b/test1/test1.cpp
new file mode 100644
index 0000000..05da6db
--- /dev/null
+++ b/test1/test1.cpp
@@ -0,0 +1,7 @@
+ #include <iostream>
+ using namespace std;
+
+ int main()
+ {
+     cout << "Hello World!\n";
+ }
diff --git a/test1/test1.sln b/test1/test1.sln
new file mode 100644
index 0000000..278d0a9
--- /dev/null
+++ b/test1/test1.sln
@@ -0,0 +1,31 @@
+
```

Рисунок 14 – Информация о созданном теге

4. Делаю 3 тега и 7 коммитов.

```

C:\Users\Admin\Desktop\git\Forprograms1.2>git status
On branch main
Your branch is ahead of 'origin/main' by 7 commits.
  (use "git push" to publish your local commits)

nothing to commit, working tree clean

C:\Users\Admin\Desktop\git\Forprograms1.2>git tag
logt
v1.1
v1.2
v1.3

C:\Users\Admin\Desktop\git\Forprograms1.2>

```

Рисунок 15 – Выполненное задание (3 тега и 7 коммитов)

5. Просмотреть историю (журнал) хранилища командой `git log`. Например, с помощью команды `git log --graph --pretty=oneline --abbrev-commit`. Добавить скриншот консоли с выводом в отчет по лабораторной работе.

```

C:\Users\Admin\Desktop\git\Forprograms1.2>git log --graph --pretty=oneline --abbrev-commit
* 11ae0dc (HEAD -> main) test1
* 693d682 (tag: v1.3, tag: logt) test1
* 7105a55 test1
* 71bcf69 test1
* a505f98 (tag: v1.2) test1
* 1bed8ca test1
* 53e15c3 test1
* 63d23e2 (tag: v1.1, origin/main, origin/HEAD) Update .gitignore
* 90be20c Create LICENSE
* 4090fbb Update README.md
* 411cfba Initial commit

```

Рисунок 16 – Вывод в консоль

6. Просмотреть содержимое коммитов командой `git show <ref>`, где `<ref>`:

HEAD: последний коммит;

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

b34a0e: коммит с указанным хэшем.

Отобразите результаты работы этих команд в отчете.

```

C:\Users\Admin\Desktop\git\Forprograms1.2>git show Head
commit 11ae0dcbfd6cb80b7ad22eaeef153a6de1c8901c (HEAD -> main)
Author: ItsMyLife1337 <itsmylife1337@yandex.ru>
Date:   Fri Mar 11 22:25:52 2022 +0300

    test1

diff --git a/test1/test1.cpp b/test1/test1.cpp
index 9c79d5f..bd95997 100644
--- a/test1/test1.cpp
+++ b/test1/test1.cpp
@@ -10,4 +10,7 @@ int main()
     cout << "Bay!\n";
     cout << "Bay!\n";
     cout << "Bay!\n";
+
+    cout << "Bay!\n";
+
 }

C:\Users\Admin\Desktop\git\Forprograms1.2>

```

Рисунок 17 – Вывод в консоль команды git show Head

```

C:\Users\Admin\Desktop\git\Forprograms1.2>git show Head~1
commit 693d68243f197cf7100d4e55ab38c5e53d0758a1 (tag: v1.3, tag: logt)
Author: ItsMyLife1337 <itsmylife1337@yandex.ru>
Date:   Fri Mar 11 22:23:39 2022 +0300

    test1

diff --git a/test1/test1.cpp b/test1/test1.cpp
index 7455cde..9c79d5f 100644
--- a/test1/test1.cpp
+++ b/test1/test1.cpp
@@ -9,4 +9,5 @@ int main()
     cout << "Привет!\n";
     cout << "Bay!\n";
     cout << "Bay!\n";
+    cout << "Bay!\n";
+
 }

C:\Users\Admin\Desktop\git\Forprograms1.2>

```

Рисунок 18 – Вывод в консоль команды git show Head~1

```

C:\Users\Admin\Desktop\git\Forprograms1.2>git show a505f98
commit a505f9848e90945ada3b096b9479e46643243477 (tag: v1.2)
Author: ItsMyLife1337 <itsmylife1337@yandex.ru>
Date:   Fri Mar 11 22:16:52 2022 +0300

    test1

diff --git a/test1/test1.cpp b/test1/test1.cpp
index 52f6660..c6928f8 100644
--- a/test1/test1.cpp
+++ b/test1/test1.cpp
@@ -6,4 +6,5 @@ int main()
     setlocale(LC_ALL, "Russian");
     cout << "Hello World!\n";
     cout << "Привет!\n";
+    cout << "Привет!\n";
+
 }

C:\Users\Admin\Desktop\git\Forprograms1.2>

```

Рисунок 19 – Вывод в консоль коммита с указанным хэшем



7. Освоить возможность отката к данной версии.

7.1. Удалите весь код из одного из файлов программы репозитория, например main.cpp, и сохраните этот файл.

7.2. Удалите все несохраненные изменения в файле командой: `git checkout -- <имя_файла>`, например `git checkout -- main.cpp`.

7.3. Повторите пункт 10.1 и сделайте коммит.

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

Сделайте выводы об изменении содержимого выбранного Вами файла программы после выполнения пунктов 7.1–7.4. Отрадите эти выводы в отчете.



Рисунок 20 – Изменил программу(удалил код) и сохранил изменения

```
c:\Users\Admin\Desktop\git\Forprograms1.2>git checkout -- test1/test1.cpp
c:\Users\Admin\Desktop\git\Forprograms1.2>git add test1/test1.cpp
c:\Users\Admin\Desktop\git\Forprograms1.2>git commit -m "test1"
c:\Users\Admin\Desktop\git\Forprograms1.2>git add test1
c:\Users\Admin\Desktop\git\Forprograms1.2>git commit -m "test1"
[main 3e64c29] test1
1 file changed, 1 insertion(+), 16 deletions(-)
c:\Users\Admin\Desktop\git\Forprograms1.2>
```

Рисунок 21 – Сделал коммит после внесения изменений

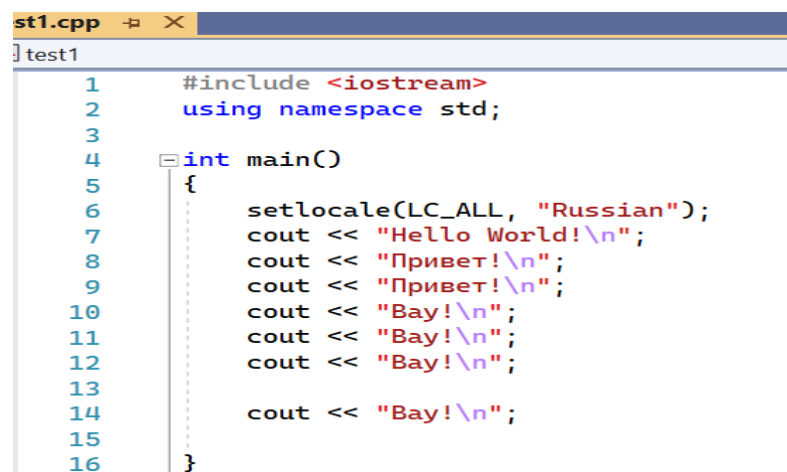
```
c:\Users\Admin\Desktop\git\Forprograms1.2>git status
On branch main
Your branch is ahead of 'origin/main' by 1 commit.
  (use "git push" to publish your local commits)

nothing to commit, working tree clean

c:\Users\Admin\Desktop\git\Forprograms1.2>git reset --hard Head~1
HEAD is now at 11ae0dc test1

c:\Users\Admin\Desktop\git\Forprograms1.2>
```

Рисунок 22 – Вернулся к предыдущей версии

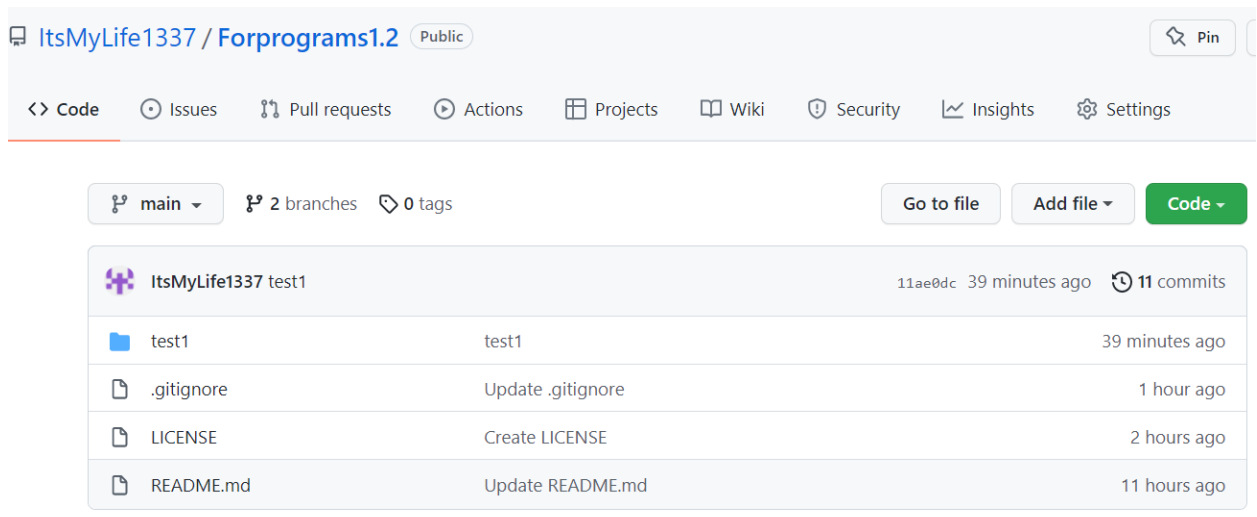


```
st1.cpp
test1
1  #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6      setlocale(LC_ALL, "Russian");
7      cout << "Hello World!\n";
8      cout << "Привет!\n";
9      cout << "Привет!\n";
10     cout << "Bay!\n";
11     cout << "Bay!\n";
12     cout << "Bay!\n";
13
14     cout << "Bay!\n";
15
16 }
```

Рисунок 23 – Вернулся к исходной программе

Выводы: освоил возможности отката к конкретным версиям программ.

## 8. Зафиксируйте изменения.



ItsMyLife1337 / Forprograms1.2 Public

<> Code Issues Pull requests Actions Projects Wiki Security Insights Settings

main 2 branches 0 tags Go to file Add file Code

File	Commit	Time
test1	test1	39 minutes ago
.gitignore	Update .gitignore	1 hour ago
LICENSE	Create LICENSE	2 hours ago
README.md	Update README.md	11 hours ago

Рисунок 24 – Зафиксировал изменения на удалённом репозитории

- Добавьте отчет по лабораторной работе в формате PDF в папку doc репозитория. Зафиксируйте изменения.

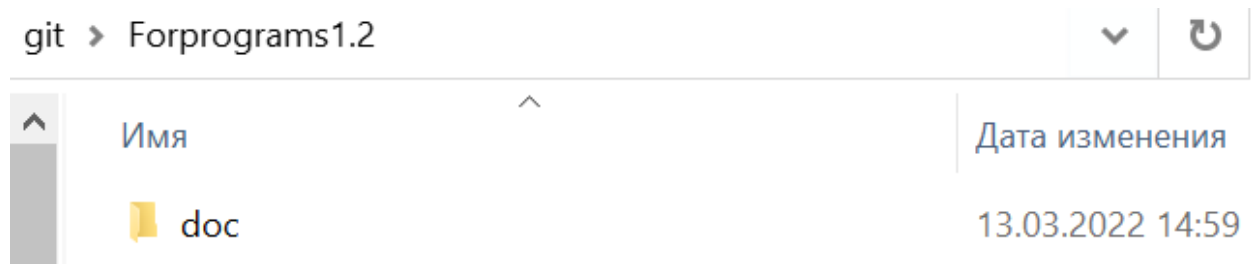


Рисунок 25 – Создал папку doc в локальном репозитории

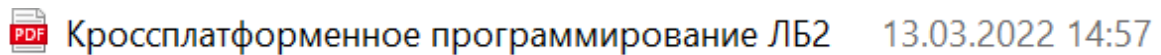


Рисунок 26 – Сохранил отчёт в папку doc

### Вопросы для защиты работы.

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

Наиболее простой и в то же время мощный инструмент для этого — команда `git log`. По умолчанию, без аргументов, `git log` выводит список коммитов созданных в данном репозитории в обратном хронологическом порядке. То есть самые последние коммиты показываются первыми.

Одна из опций, когда вы хотите увидеть сокращенную статистику для каждого коммита, вы можете использовать опцию `—stat`.

Вторая опция (одна из самых полезных аргументов) является `-p` или `--patch`, который показывает разницу (выводит патч), внесенную в каждый коммит. Так же вы можете ограничить количество записей в выводе команды; используйте параметр `-2` для вывода только двух записей (пример команды `git log -p -2`).

Третья действительно полезная опция это `--pretty`. Она меняет формат вывода. Существует несколько встроенных вариантов отображения. Опция `oneline` выводит каждый коммит в одну строку, что может быть очень удобным если вы просматриваете большое количество коммитов. К тому же, опции `short`, `full` и `fuller` делают вывод приблизительно в том же формате, но с меньшим или большим количеством информации соответственно.

Опция	Описания вывода
%H	Хеш коммита
%h	Сокращенный хеш коммита
%T	Хеш дерева
%t	Сокращенный хеш дерева
%P	Хеш родителей
%p	Сокращенный хеш родителей
%an	Имя автора
%ae	Электронная почта автора
%ad	Дата автора (формат даты можно задать опцией --date=option)
%ar	Относительная дата автора
%cn	Имя коммитера
%ce	Электронная почта коммитера
%cd	Дата коммитера
%cr	Относительная дата коммитера
%s	Содержание

Наиболее интересной опцией является `format`, которая позволяет указать формат для вывода информации. Особенно это может быть полезным, когда вы хотите сгенерировать вывод для автоматического анализа — так как вы указываете формат явно, он не будет изменен даже после обновления Git.

Для опции `git log --pretty=format` существуют различного рода опции для изменения формата отображения.

## 2. Как ограничить вывод при просмотре истории коммитов?

Для ограничения может использоваться функция `git log <n>`, где `n` число записей.

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

```
git log --since=2.weeks
```

Это команда работает с большим количеством форматов — вы можете указать определенную дату вида 2008-01-15 или же относительную дату, например 2 years 1 day 3 minutes ago.

Также вы можете фильтровать список коммитов по заданным параметрам. Опция --author дает возможность фильтровать по автору коммита, а опция --grep (показывает только коммиты, сообщение которых содержит указанную строку) искать по ключевым словам в сообщении коммита. Функция -S показывает только коммиты, в которых изменение в коде повлекло за собой добавление или удаление указанной строки.

### 3. Как внести изменения в уже сделанный коммит?

Внести изменения можно с помощью команды `git commit --amend`

Эта команда берёт индекс и применяет его к последнему коммиту. Если после последнего коммита не было никаких проиндексированных изменений (например, вы запустили приведённую команду сразу после предыдущего коммита), то состояние проекта будет абсолютно таким же и всё, что мы изменим, это комментарий к коммиту.

Для того, чтобы внести необходимые изменения - нам нужно проиндексировать их и выполнить команду `git commit --amend`.

```
git commit -m 'initial commit'
```

```
git add forgotten_file
```

```
git commit --amend
```

Эффект от выполнения этой команды такой, как будто мы не выполнили предыдущий коммит, а еще раз выполнили команду `git add` и выполнили коммит.

### 4. Как отменить индексацию файла в Git?

Например, вы изменили два файла и хотите добавить их в разные коммиты, но случайно выполнили команду `git add *` и добавили в индекс оба. Как исключить из индекса один из них? Команда `git status` напомним вам:

Прямо под текстом «Changes to be committed» говорится: используйте `git reset HEAD <file>` для исключения из индекса.

#### 5. Как отменить изменения в файле?

С помощью команды `git checkout -- <file>`.

#### 6. Что такое удаленный репозиторий Git?

Удалённый репозиторий это своего рода наше облако, в которое мы сохраняем те или иные изменения в нашей программе/коде/файлах.

#### 7. Как выполнить просмотр удаленных репозитория данного локального репозитория?

Для того, чтобы просмотреть список настроенных удалённых репозитория, необходимо запустить команду `git remote`.

Также можно указать ключ `-v`, чтобы просмотреть адреса для чтения и записи, привязанные к репозиторию. Пример: `git remote -v`

#### 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?

Если необходимо получить изменения, которые есть у Пола, но нету у вас, вы можете выполнить команду `git fetch <Название репозитория>`.

Важно отметить, что команда `git fetch` забирает данные в ваш локальный репозиторий, но не сливает их с какими-либо вашими наработками и не модифицирует то, над чем вы работаете в данный момент. Вам необходимо вручную слить эти данные с вашими, когда вы будете готовы.

Если ветка настроена на отслеживание удалённой ветки, то вы можете использовать команду `git pull` чтобы автоматически получить изменения из удалённой ветки и слить их со своей текущей. Выполнение `git pull`, как правило, извлекает (`fetch`) данные с сервера, с которого вы изначально клонировали, и автоматически пытается слить (`merge`) их с кодом, над которым вы в данный момент работаете.

Чтобы отправить изменения на удалённый репозиторий необходимо отправить их в удалённый репозиторий. Команда для этого действия простая: `git push <remote-name> <branch-name>`.

#### 10. Как выполнить просмотр удаленного репозитория?

Для просмотра удалённого репозитория, можно использовать команду `git remote show <remote>`.

#### 11. Каково назначение тэгов Git?

Теги - это ссылки указывающие на определённые версии кода/написанной программы. Они удобно чтобы в случае чего вернуться к нужному моменту. Также при помощи тегов можно помечать важные моменты.

#### 12. Как осуществляется работа с тэгами Git?

Просмотреть наличие тегов можно с помощью команды: `git tag`.

А назначить (указать, добавить тег) можно с помощью команды `git tag -a v1.4(версия изначальная) -m "Название"`.

С помощью команды `git show` вы можете посмотреть данные тега вместе с коммитом: `git show v1.4`.

Отправка тегов, по умолчанию, команда `git push` не отправляет теги на удалённые сервера. После создания тега нужно отправлять явно на удалённый сервер. Процесс аналогичен отправке веток — достаточно выполнить команду `git push origin <tagname>`. Для отправки всех тегов можно использовать команду `git push origin tags`.

Для удаления тега в локальном репозитории достаточно выполнить команду `git tag -d <tagname>`. Например, удалить созданный ранее лёгковесный тег можно следующим образом: `git tag -d v1.4-lw`

Для удаления тега из внешнего репозитория используется команда `git push origin --delete <tagname>`.

Если вы хотите получить версии файлов, на которые указывает тег, то вы можете сделать `git checkout` для тега пример: `git checkout -b version2 v2.0.0`.

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

`Git fetch --prune` команда получения всех изменений с репозитория GitHub.

В команде `git push --prune` удаляет удаленные ветки, у которых нет локального аналога.

**Вывод:** исследовал базовые возможности системы контроля версий `git` для работы с локальными репозиториями. Также, благодаря созданию тегов и пункту 7 лабораторной работы после изменения файлов освоил возможность отката к заданной версии.