

Федеральное государственное автономное образовательное учреждение
высшего образования «Северо-Кавказский федеральный университет»

Институт цифрового развития

ОТЧЕТ ПО ДИСЦИПЛИНЕ
ОСНОВЫ ПРОГРАММНОЙ ИНЖЕНЕРИИ

Выполнил:

Турклиев Владимир

2 курс, группа ПИЖ-б-о-21-1

Принял:

Воронкин Роман Александрович

Ставрополь, 2022 г.

ВЫПОЛНЕНИЕ


1) Создание репозитория.

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Owner *

Repository name *


 vegas007gof ▾

 /


lab1 ✓

Great repository names are short and memorable. Need inspiration? How about [symmetrical-winner?](#)

Description (optional)

☒  Public

Anyone on the internet can see this repository. You choose who can commit.

☐  Private

You choose who can see and commit to this repository.

Initialize this repository with:

Skip this step if you're importing an existing repository.

☐ Add a README file

This is where you can write a long description for your project. [Learn more.](#)

☒ Add .gitignore


Choose which files not to track from a list of templates. [Learn more.](#)

.gitignore template: C++ ▾

☒ Choose a license

A license tells others what they can and can't do with your code. [Learn more.](#)

License: Лицензия MIT ▾

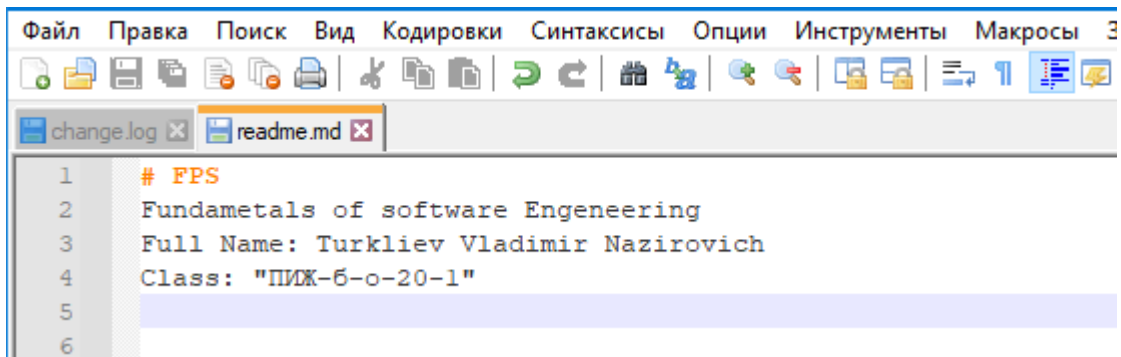
This will set  main as the default branch. Change the default name in your [settings](#).

Create repository

2) Клонирование репозитория на компьютер

```
C:\Users\Vova>cd desktop
C:\Users\Vova\Desktop>cd учеба
C:\Users\Vova\Desktop\учеба>cd основы ии
C:\Users\Vova\Desktop\учеба\основы ии>cd lab1
C:\Users\Vova\Desktop\учеба\основы ии\lab1>git clone https://github.com/vegas007gof/lab1.git
Cloning into 'lab1'...
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (4/4), done.
C:\Users\Vova\Desktop\учеба\основы ии\lab1>
```

3) Добавление информации в «readme.md»



The screenshot shows a code editor with a menu bar (Файл, Правка, Поиск, Вид, Кодировки, Синтаксисы, Опции, Инструменты, Макросы) and a toolbar. The 'readme.md' file is open, showing the following content:

```
1 # FPS
2 Fundamentals of software Engineering
3 Full Name: Turkliiev Vladimir Nazirovich
4 Class: "ПИЖ-б-о-20-1"
5
6
```

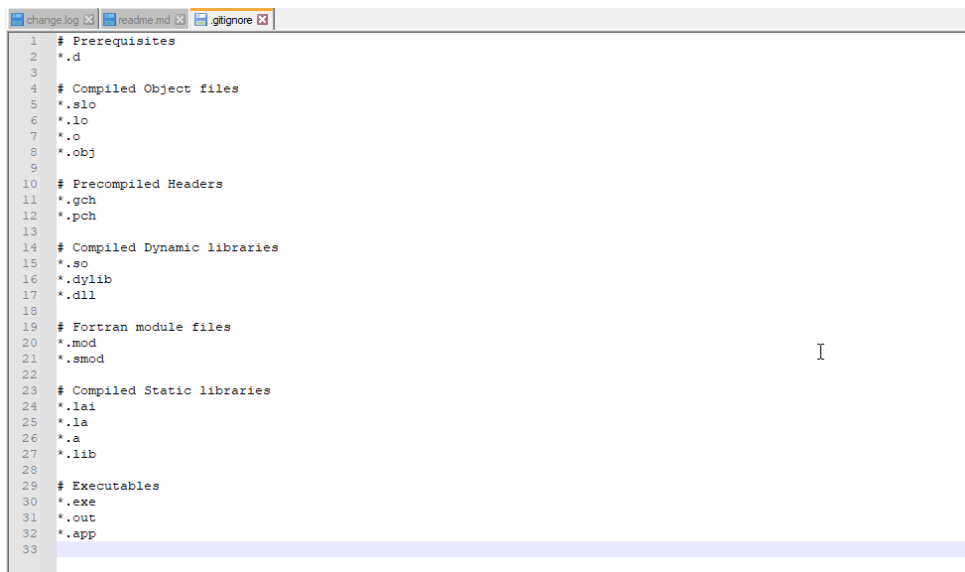
4) Делаем commit

```
C:\Users\Vova\Desktop\учеба\основы ии\lab1\lab1>git add README.md
C:\Users\Vova\Desktop\учеба\основы ии\lab1\lab1>git add readme.md
C:\Users\Vova\Desktop\учеба\основы ии\lab1\lab1> git commit -m "hope"
[main 569e46c] hope
 1 file changed, 4 insertions(+)

C:\Users\Vova\Desktop\учеба\основы ии\lab1\lab1> git push
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 8 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 347 bytes | 347.00 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/vegas007gof/lab1.git
 a30ba4f..569e46c main -> main

C:\Users\Vova\Desktop\учеба\основы ии\lab1\lab1>
```

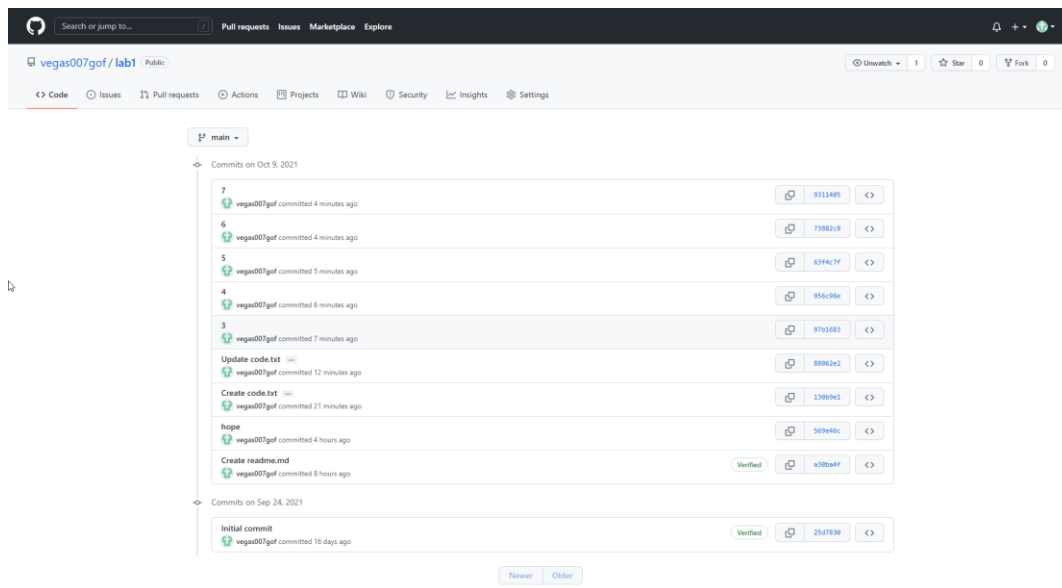
5) Файл .gitignore



The screenshot shows a code editor with a menu bar and a toolbar. The '.gitignore' file is open, showing the following content:

```
1 # Prerequisites
2 *.d
3
4 # Compiled Object files
5 *.slo
6 *.lo
7 *.o
8 *.obj
9
10 # Precompiled Headers
11 *.gch
12 *.pch
13
14 # Compiled Dynamic libraries
15 *.so
16 *.dylib
17 *.dll
18
19 # Fortran module files
20 *.mod
21 *.smod
22
23 # Compiled Static libraries
24 *.lai
25 *.la
26 *.a
27 *.lib
28
29 # Executables
30 *.exe
31 *.out
32 *.app
33
```

6) История коммитов



Ответы на контрольные вопросы:

Ответы на вопросы:

1. Что такое СКВ и каково ее назначение?

Система контроля версий (СКВ) — это система, регистрирующая изменения в одном или нескольких файлах с тем, чтобы в дальнейшем

была возможность вернуться к определённым старым версиям этих файлов.

2. В чем недостатки локальных и централизованных СКВ?

Одним из серьезных недостатков является единая точка отказа. В случае ее отказа теряется вся проделанная работа. Также затруднен доступ нескольких людей к одному проекту в случае с локальными СКВ

3. К какой СКВ относится Git?

Распределённой

4. В чем концептуальное отличие Git от других СКВ?

Основное отличие Git от любой другой СКВ – это подход к работе со своими данными. Концептуально, большинство других систем хранят информацию в виде списка изменений в файлах. Эти системы представляют хранимую информацию в виде набора файлов и изменений, сделанных в каждом файле, по времени (обычно это называют контролем версий, основанным на различиях). Git не хранит и не обрабатывает данные таким способом. Вместо этого, подход Git к хранению данных больше похож на набор снимков миниатюрной файловой системы. Каждый раз, когда вы делаете коммит, то есть сохраняете состояние своего проекта в Git, система запоминает, как выглядит каждый файл в этот момент, и сохраняет ссылку на этот снимок. Для увеличения эффективности, если файлы не были изменены, Git не запоминает эти файлы вновь, а только создаёт ссылку на предыдущую версию идентичного файла, который уже сохранён. Git представляет свои данные как, скажем, поток снимков.

5. Как обеспечивается целостность хранимых данных в Git?

В Git для всего вычисляется хеш-сумма, и только потом происходит сохранение. В дальнейшем обращение к сохранённым объектам происходит по этой хеш-сумме. Это значит, что невозможно изменить содержимое файла или директории так, чтобы Git не узнал об этом. Данная функциональность встроена в Git на низком уровне и является неотъемлемой частью его философии. Вы не потеряете информацию во время её передачи и не получите повреждённый файл без ведома Git. Механизм, которым пользуется Git при вычислении хеш-сумм, называется SHA-1 хеш. Это строка длиной в 40

шестнадцатеричных символов (0-9 и a-f), она вычисляется на основе содержимого файла или структуры каталога.

6. В каких состояниях могут находиться файлы в Git? Как связаны эти состояния?

У Git есть три основных состояния, в которых могут находиться ваши файлы: зафиксированное (committed), изменённое (modified) и подготовленное (staged). Зафиксированный значит, что файл уже сохранён в вашей локальной базе. К изменённым относятся файлы, которые поменялись, но ещё не были зафиксированы. Подготовленные файлы — это изменённые файлы, отмеченные для включения в следующий коммит.

Мы подошли к трём основным секциям проекта Git: Git-директория (Git directory), рабочая директория (working directory) и область подготовленных файлов (staging area). Git-директория — это то место, где Git хранит метаданные и базу объектов вашего проекта. Это самая важная часть Git, и это та часть, которая копируется при клонировании репозитория с другого компьютера. Рабочая директория является снимком версии проекта. Файлы распаковываются из сжатой базы данных в Git-директории и располагаются на диске, для того чтобы их можно было изменять и использовать. Область подготовленных файлов — это файл, обычно располагающийся в вашей Git-директории, в нём содержится информация о том, какие изменения попадут в следующий коммит. Эту область ещё называют “индекс”, однако называть её stage-область также общепринято.

7. Что такое профиль пользователя в GitHub?

Страница, на которой можно указать информацию о себе.

8. Какие бывают репозитории в GitHub?

Общедоступные и приватные.

9. Укажите основные этапы модели работы с GitHub.

- 1) Создание аккаунта.
- 2) Создание репозитория.
- 3) Клонирование репозитория.

4) Локальное изменение содержимого.

5) Отправка изменений в удаленный репозиторий с помощью Git.

10. Как осуществляется первоначальная настройка Git после установки?

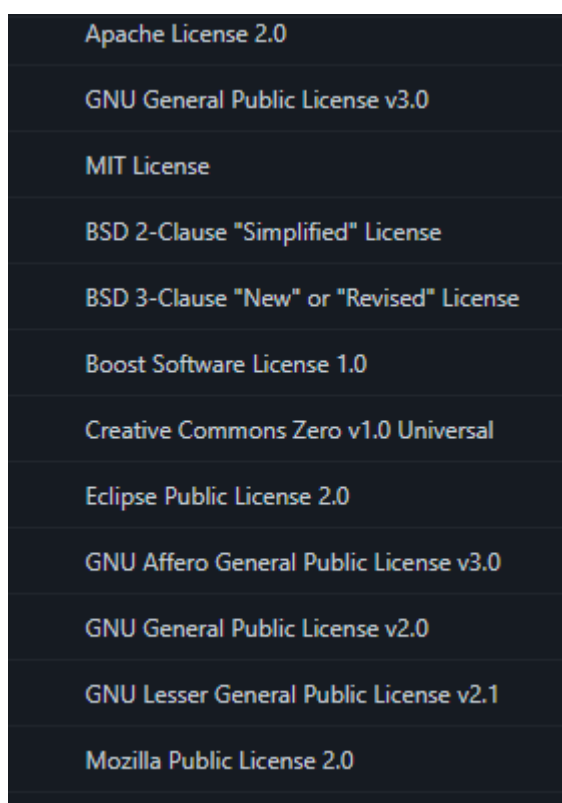
Чтобы убедиться, что Git установлен, нужно написать в консоли команду «git version». Затем нужно добавить в Git имя пользователя и почту с GitHub. Перед первой отправкой на сервер необходимо передать локальную ветку с помощью следующей команды: `git push --set-upstream origin edit-readme`.

11. Опишите этапы создания репозитория в GitHub.

В правом верхнем углу нажать на кнопку создания нового репозитория. Затем указать его название, описание и выбрать необходимые предустановки, такие как: вид репозитория (открытый/закрытый), создание файла `.gitignore` и выбор лицензии.

12. Какие типы лицензий поддерживаются GitHub при создании репозитория?

Ответ представлен на рисунке ниже



13. Как осуществляется клонирование репозитория GitHub? Зачем нужно клонировать репозиторий?

После создания репозитория его необходимо клонировать на ваш компьютер. Для этого на странице репозитория необходимо найти кнопку Clone или Code и щелкнуть по ней, чтобы отобразить адрес репозитория для клонирования. Клонирование выполняется для внесения в репозиторий локальных изменений и, наконец, запроса на обновление файлов в удаленном репозитории.

14. Как проверить состояние локального репозитория Git?

С помощью команды «git status».

15. Как изменяется состояние локального репозитория Git после выполнения следующих операций: добавления/изменения файла в локальный репозиторий Git; добавления нового/измененного файла под версионный контроль с помощью команды git add; фиксации(коммита) изменений с помощью команды git commit и отправки изменений на сервер спомощью команды git push?

При добавлении/изменении файлов они помечаются как «modified». При добавлении под версионный контроль файл помечается как «new file». При фиксации изменений статус меняется на «... ahead on 1 commit», после отправки статус выдает: «your branch is up to date».

16. У Вас имеется репозиторий на GitHub и два рабочих компьютера, с помощью которых Вы можете осуществлять работу над некоторым проектом с использованием этого репозитория. Опишите последовательность команд, с помощью которых оба локальных репозитория, связанных с репозиторием GitHub будут находиться в синхронизированном состоянии. Примечание: описание необходимо начать с команды git clone.

Необходимо клонировать исходный репозиторий нв каждый из компьютеров с помощью команды «git clone», на каждом компьютере после внесения локальных изменений нужно добавлять их под версионный контроль (git add), делать коммит изменений (git commit –m), отправлять изменения на сервер (git push), для получения новых версий файлов из репозитория необходимо использовать команду «git pull».

17. GitHub является не единственным сервисом, работающим с Git. Какие сервисы еще Вам известны? Приведите сравнительный анализ одного из таких сервисов с GitHub.

GitLab, SourceForge, BitBucket, Launchpad, Apache Allura, Cloud Source, AWS code commit, FogCreek/DevHub, BeanStalk, GitKraken.

GitHub имеет удобный интерфейс, прост в использовании, но в отличие от sourceforge работает только с Git.

18. Интерфейс командной строки является не единственным и далеко не самым удобным способом работы с Git. Какие Вам известны программные средства с графическим интерфейсом пользователя для работы с Git? Приведите как реализуются описанные в лабораторной работе операции Git с помощью одного из таких программных средств.

Fork, Tower, Sourcetree, SmartGit, GitKraken и т.д. В таких программах действия, выполняемые с помощью консольных команд, представлены в пользовательском интерфейсе, что значительно упрощает знакомство с Git и дальнейшее использование.