

# Лабораторная работа №2

## Отчёт к лабораторной работе

Зайцева Анна Дмитриевна

### Table of Contents

### Цель работы

Цель работы — изучить идеологию и применение средств контроля версий, освоить умения по работе с git.

### Задание

- Изучить идеологию и применение средств контроля версий.
- Освоить умения по работе с git.

### Выполнение лабораторной работы

1) Создаём учётную запись на <https://github.com>. (Рис. [-@fig:001]):

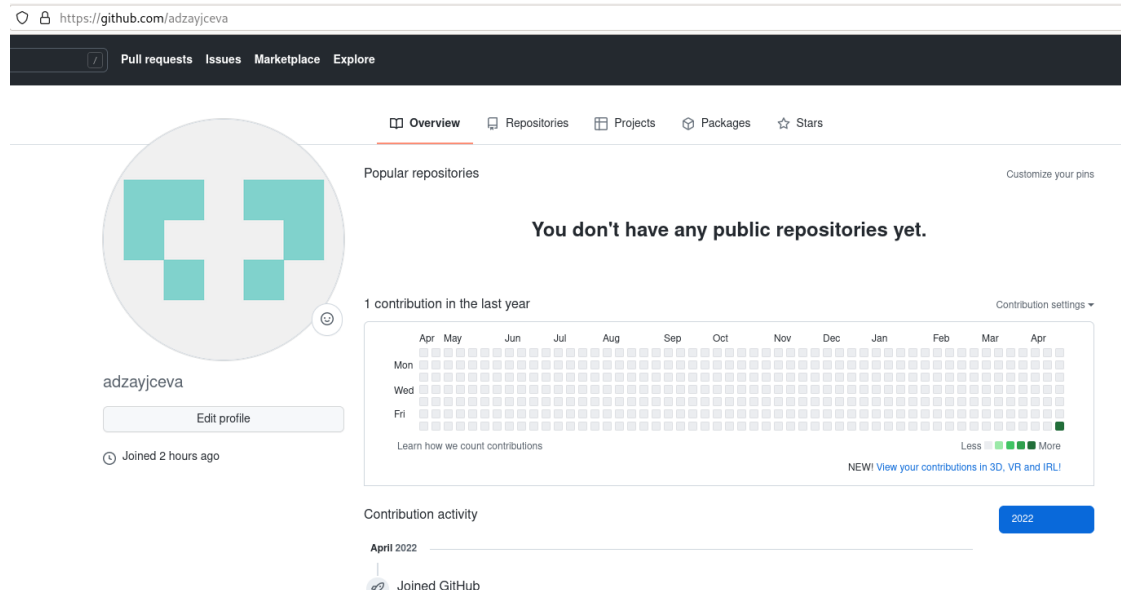


Рис. 1

2) Произведём базовую настройку git. Задаём имя и email владельца репозитория, после чего настроим utf-8 в выводе сообщений git. (Рис. [-@fig:002]):

```
[adzayjceva@adzayjceva ~]$ git config --global user.name"adzayjceva"
[adzayjceva@adzayjceva ~]$ git config -- global user.email"zajceva186@gmail.com"
fatal: not in a git directory
[adzayjceva@adzayjceva ~]$ git config --global user.email"zajceva186@gmail.com"
[adzayjceva@adzayjceva ~]$ git config --global core.quotepath false
```

Рис. 2

- 3) Создадим новый ключ на github. (*ssh-keygen -C"adzayjcevazajceva186@gmail.com"*) и привяжем его к компьютеру через консоль. Нам нужно загрузить сгенерённый нами заранее открытый ключ. Заходим на наш github → «settings» → «ssh keys» → «add key». Копируем из локальной консоли ключ в буфер обмена (*cat ~/.ssh/id\_rsa.pub | xclip -sel clip*) и вставляем ключ в появившееся на сайте поле. (Рис. [-@fig:003])

```

[adzayjceva@adzayjceva ~]$ ssh-keygen -C"adzayjceva<zajceval86@gmail.com>"
Generating public/private rsa key pair.
Enter file in which to save the key (/home/adzayjceva/.ssh/id_rsa):
/home/adzayjceva/.ssh/id_rsa already exists.
Overwrite (y/n)? y
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/adzayjceva/.ssh/id_rsa
Your public key has been saved in /home/adzayjceva/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:8FjzP0lmpSmqINDtPTH+dfUKUCam9UBffMi0KwsnzZs adzayjceva<zajceval86@gmail.com>
The key's randomart image is:
+---[RSA 3072]-----+
|      . +o. |
|      . . +..|
|      . o = + ..|
|      . . = B X .|
|      . . . + S O B o |
|      . . o o . X * .|
|      . . . + . B E .|
|      . . + = . . .|
|      . . . . .|
+-----[SHA256]-----+
[adzayjceva@adzayjceva ~]$ cat ~/.ssh/id_rsa.pub | xclip -sel clip
bash: xclip: command not found...
Install package 'xclip' to provide command 'xclip'? [N/y] y

* Waiting in queue...
* Loading list of packages....
The following packages have to be installed:
xclip-0.13-15.git11cba61.fc35.x86_64 Command line clipboard grabber
Proceed with changes? [N/y] y

* Waiting in queue...
* Waiting for authentication...
* Waiting in queue...
* Downloading packages...
* Requesting data...
* Testing changes...
* Installing packages...

[adzayjceva@adzayjceva ~]$ cat ~/.ssh/id_rsa.pub | xclip -sel clip

```


Рис. 3

Далее мы создадим и подключим репозиторий к github. На сайте заходим в раздел «repositories» → «new» → создаём новый репозиторий (имя: «laboratory» и добавим файл README) (Рис. [-@fig:004]) И скопируем ссылку на репозиторий в консоль для дальнейшей работы с файлами. (Рис. [-@fig:005])

## Create a new repository

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

Owner \*

 adzayjceva ▾

Repository name \*

/ laboratory ✓

Great repository names are short and memorable. Need inspiration? How about **improved-giggle**?

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: None ▾

Рис. 4

```
[adzayjceva@adzayjceva ~]$ git clone https://github.com/adzayjceva/laboratory.git
Клонирование в «laboratory»...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Получение объектов: 100% (3/3), готово.
[adzayjceva@adzayjceva ~]$
```

Рис. 5

- 4) Алгоритм создания структуры каталога через консоль описан в лабораторной, но будет легче создать репозиторий на github, после этого работать с каталогом и папками через консоль (перед этим нужно скопировать ссылку на репозиторий в консоль в формате https или ssh). Перед созданием файлов зайдём в наш репозиторий Рис. [-@fig:006]):

```
[adzayjceva@adzayjceva ~]$ cd laboratory
[adzayjceva@adzayjceva laboratory]$ ls
README.md
[adzayjceva@adzayjceva laboratory]$
```

Рис. 6

После этого уже можно создавать файлы (Рис. [-@fig:007]):

```
[adzayjceva@adzayjceva laboratory]$ mkdir 2021-2022
[adzayjceva@adzayjceva laboratory]$ cd 2021-2022
[adzayjceva@adzayjceva 2021-2022]$ mkdir OS
[adzayjceva@adzayjceva 2021-2022]$ cd OS
[adzayjceva@adzayjceva OS]$ mkdir lab02
[adzayjceva@adzayjceva OS]$ cd lab02
[adzayjceva@adzayjceva lab02]$ cd ..
[adzayjceva@adzayjceva OS]$ cd lab02
[adzayjceva@adzayjceva lab02]$
```

Рис. 7

- 5) Добавим первый коммит и выложим его на github. Для правильного размещения первого коммита нам необходимо добавить команду *git add .*, после чего с помощью команды *git commit -am "first commit"* выкладываем коммит (Рис. [-@fig:008]):

```
[adzayjceva@adzayjceva lab02]$ git config --global user.email "zajceva186@gmail.com"
[adzayjceva@adzayjceva lab02]$ git config --global user.name "adzayjceva"
[adzayjceva@adzayjceva lab02]$ ls
2.txt README.md
[adzayjceva@adzayjceva lab02]$ git add .
[adzayjceva@adzayjceva lab02]$ git commit -am "first commit"
[main 697a87e] first commit
3 files changed, 1 insertion(+)
create mode 100644 2021-2022/OS/lab02/.txt
create mode 100644 2021-2022/OS/lab02/2.txt
create mode 100644 2021-2022/OS/lab02/README.md
[adzayjceva@adzayjceva lab02]$
```

Рис. 8

- 6) Сохраняем первый коммит, используя команду *git push* (Рис. [-@fig:009]):

```
[adzayjceva@adzayjceva lab02]$ git push
Перечисление объектов: 8, готово.
Подсчет объектов: 100% (8/8), готово.
При сжатии изменений используется до 2 потоков
Сжатие объектов: 100% (3/3), готово.
Запись объектов: 100% (7/7), 477 байтов | 238.00 КиБ/с, готово.
Всего 7 (изменений 0), повторно использовано 0 (изменений 0), повторно использовано пакетов 0
To https://github.com/adzayjceva/laboratory.git
   1e693a3..697a87e  main -> main
[adzayjceva@adzayjceva lab02]$
```

Рис. 9

## 7) Первичная конфигурация:

### 1. Добавим файл лицензии (Рис. [-@fig:010]):

```
[adzayjceva@adzayjceva lab02]$ wget https://creativecommons.org/licenses/by/4.0/legalcode.txt -O LICENCE
--2022-04-23 22:49:46-- https://creativecommons.org/licenses/by/4.0/legalcode.txt
Распознаётся creativecommons.org (creativecommons.org)... 104.20.151.16, 172.67.34.140, 104.20.150.16, ...
Подключение к creativecommons.org (creativecommons.org)|104.20.151.16|:443... соединение установлено.
HTTP-запрос отправлен. Ожидание ответа... 200 OK
Длина: нет данных [text/plain]
Сохранение в: «LICENCE»

LICENCE [ <=>

2022-04-23 22:49:47 (7,79 MB/s) - «LICENCE» сохранён [18657]
[adzayjceva@adzayjceva lab02]$
```

Рис. 10

### 2. Добавим шаблон игнорируемых файлов. Просмотрим список имеющихся шаблонов (на скрине список неполный) (Рис. [-@fig:011]):

```
[adzayjceva@adzayjceva lab02]$ curl -L -s https://www.gitignore.io/api/list
1c,1c-bitrix,a-frame,actionsript,ada
adobe,advancedinstaller,adventuregamestudio,agda,al
alteraquartusii,altium,amplify,android,androidstudio
angular,anjuta,ansible,ansibletower,apachecordova
apachehadoop,appbuilder,appcelerator titanium,appcode,appcode+all
appcode+iml,appengine,aptanastudio,arcanist,archive
archives,archlinuxpackages,aspnetcore,assembler,ate
atmelstudio,ats,audio,automationstudio,autotools
autotools+strict,awr,azurefunctions,azurite,backup
ballerina,basercms,basic,batch,bazaar
bazel,bitrise,bitrix,bittorrent,blackbox
bloop,bluej,bookdown,bower,bricxcc
buck,c,c++,cake,cakephp
cakephp2,cakephp3,calabash,carthage,certificates
ceylon,cfwheels,chefcookbook,chocolatey,circuitpython
clean,clion,clion+all,clion+iml,clojure
cloud9,cmake,cocoapods,cocos2dx,cocoscreator
codeblocks,codecomposerstudio,codeigniter,codeio,codekit
codesniffer,coffeescript,commonlisp,compodoc,composer
compressed,compressedarchive,compression,conan,concrete5
coq,cordova,craftcms,crashlytics,crbasic
crossbar,crystal,cs-cart,csharp,cuda
cvs,cypressio,d,dart,darteditor
data,database,datarecovery,dbeaver,defold
delphi,dframe,diff,direnv,diskimage
django,dm,docfx,docpress,docz
dotenv,dotfilessh,dotnetcore,dotsettings,dreamweaver
dropbox,drupal,drupal7,drupal8,e2studio
eagle,easybook,eclipse,eiffelstudio,elasticbeanstalk
elisp,elixir,elm,emacs,ember
ensime,episerver,erlang,espresso,executable
exercism,expressionengine,extjs,fancy,fastlane
finale,firebase,flashbuilder,flask,flatpak
flex,flexbuilder,floobits,flutter,font
fontforge,forcedotcom,forgegradle,fortran,freecad
freepascal,fsharp,fuelphp,fusetools,games
gcov,genero4gl,geth,ggts,gis
git,gitbook,go,godot,goland
goodsync,gpg,gradle,grails,greenfoot
groovy,grunt,gwt,haskell,helm
hexo,hol,homeassistant,homebrew,hsp
hugo,hyperledgercomposer,iar,iar_ewarm,iarembdedworkbench
idanpro,idris,igorpro,images,infer
```

Рис. 11

3. Скачиваем шаблон, например, для C. Ещё добавляем новые файлы и выполняем коммит (Рис. [-@fig:012]):

```
zendframework,zephir,zig,zsh,zukencr8000[adzayjceva@adzayjceva lab02]$ curl -L -s https://www.gitignore.io/api/c >> .gitignore
[adzayjceva@adzayjceva lab02]$ git add .
[adzayjceva@adzayjceva lab02]$ git commit -am 'Создали шаблон для C'
[main 9b54d71] Создали шаблон для C
2 files changed, 455 insertions(+)
create mode 100644 2021-2022/OS/lab02/.gitignore
create mode 100644 2021-2022/OS/lab02/LICENCE
[adzayjceva@adzayjceva lab02]$
```

Рис. 12

4. Отправляем на github (нужно сохранить все созданные шаблоны и файлы с помощью команды *git push*) (Рис. [-@fig:013]):

```
[adzayjceva@adzayjceva lab02]$ git push
Перечисление объектов: 11, готово.
Подсчет объектов: 100% (11/11), готово.
При сжатии изменений используется до 2 потоков
Сжатие объектов: 100% (5/5), готово.
Запись объектов: 100% (7/7), 6.65 КиБ | 2.22 МиБ/с, готово.
Всего 7 (изменений 0), повторно использовано 0 (изменений 0), повторно использовано пакетов 0
To https://github.com/adzayjceva/laboratory.git
697a87e..9b54d71 main -> main
[adzayjceva@adzayjceva lab02]$
```

Рис. 13

- 8) Работаем с конфигурацией git-flow

1. Установка git-flow в Fedora Linux (Рис. [-@fig:014]):

```
[adzayjceva@adzayjceva lab02]$ cd /tmp
[adzayjceva@adzayjceva tmp]$ wget --no-check-certificate -q https://raw.githubusercontent.com/peterwunderdoes/gitflow/develop/contrib/gitflow-installer.sh
[adzayjceva@adzayjceva tmp]$ chmod +x gitflow-installer.sh
chmod: невозможно получить доступ к 'gitflow-installer.sh': Нет такого файла или каталога
[adzayjceva@adzayjceva tmp]$ wget --no-check-certificate -q https://raw.githubusercontent.com/petervanderdoes/gitflow/develop/contrib/gitflow-installer.sh
[adzayjceva@adzayjceva tmp]$ chmod +x gitflow-installer.sh
[adzayjceva@adzayjceva tmp]$ sudo ./gitflow-installer.sh install stable
[sudo] пароль для adzayjceva:
## git-flow no-make installer ##
Installing git-flow to /usr/local/bin
Cloning repo from GitHub to gitflow
Клонирование в «gitflow»...
remote: Enumerating objects: 4270, done.
remote: Total 4270 (delta 0), reused 0 (delta 0), pack-reused 4270
Получение объектов: 100% (4270/4270), 1.74 МиБ | 2.26 МиБ/с, готово.
Определение изменений: 100% (2533/2533), готово.
Уже обновлено.
Ветка «master» отслеживает внешнюю ветку «master» из «origin».
Переключено на новую ветку «master»
install: создание каталога '/usr/local/share/doc'
install: создание каталога '/usr/local/share/doc/gitflow'
install: создание каталога '/usr/local/share/doc/gitflow/hooks'
'gitflow/git-flow' -> '/usr/local/bin/git-flow'
'gitflow/git-flow-init' -> '/usr/local/bin/git-flow-init'
'gitflow/git-flow-feature' -> '/usr/local/bin/git-flow-feature'
'gitflow/git-flow-bugfix' -> '/usr/local/bin/git-flow-bugfix'
'gitflow/git-flow-hotfix' -> '/usr/local/bin/git-flow-hotfix'
'gitflow/git-flow-release' -> '/usr/local/bin/git-flow-release'
'gitflow/git-flow-support' -> '/usr/local/bin/git-flow-support'
'gitflow/git-flow-version' -> '/usr/local/bin/git-flow-version'
'gitflow/gitflow-common' -> '/usr/local/bin/gitflow-common'
'gitflow/gitflow-shFlags' -> '/usr/local/bin/gitflow-shFlags'
'gitflow/git-flow-config' -> '/usr/local/bin/git-flow-config'
```

Рис. 14

2. Инициализируем git-flow через команду *git flow init -f* (префикс для ярлыков установлен в v) (Рис. [-@fig:015]):



```
[adzayjceva@adzayjceva tmp]$ git flow init -f
подсказка: Using 'master' as the name for the initial branch. This default branch name
подсказка: is subject to change. To configure the initial branch name to use in all
подсказка: of your new repositories, which will suppress this warning, call:
подсказка:
подсказка:     git config --global init.defaultBranch <name>
подсказка:
подсказка: Names commonly chosen instead of 'master' are 'main', 'trunk' and
подсказка: 'development'. The just-created branch can be renamed via this command:
подсказка:
подсказка:     git branch -m <name>
Инициализирован пустой репозиторий Git в /tmp/.git/
No branches exist yet. Base branches must be created now.
Branch name for production releases: [master]
Branch name for "next release" development: [develop]

How to name your supporting branch prefixes?
Feature branches? [feature/] v
Bugfix branches? [bugfix/]
Release branches? [release/]
Hotfix branches? [hotfix/]
Support branches? [support/]
Version tag prefix? [] v
Hooks and filters directory? [/tmp/.git/hooks]
[adzayjceva@adzayjceva tmp]$
```

Рис. 15

3. Проверим, что мы находимся на ветке develop (команда git branch) (Рис. [-@fig:016]):

```
[adzayjceva@adzayjceva tmp]$ git branch
* develop
  master
[adzayjceva@adzayjceva tmp]$
```

Рис. 16

4. Создадим релиз с версией 1.0.0 (Рис. [-@fig:017]):

```
[adzayjceva@adzayjceva tmp]$ git flow release start 1.0.0
Переключено на новую ветку «release/1.0.0»

Summary of actions:
- A new branch 'release/1.0.0' was created, based on 'develop'
- You are now on branch 'release/1.0.0'

Follow-up actions:
- Bump the version number now!
- Start committing last-minute fixes in preparing your release
- When done, run:

    git flow release finish '1.0.0'

[adzayjceva@adzayjceva tmp]$
```

Рис. 17

5. Записываем версию и добавляем в индекс (Рис. [-@fig:018]):

*echo „hello world“>hello.txt git add hello.txt git commit -am „new file“*

```
[adzayjceva@adzayjceva tmp]$ echo "1.0.0">> VERSION
[adzayjceva@adzayjceva tmp]$ git add .
warning: не удалось открыть каталог «systemd-private-880b4b7bc6f244d1a9b341b39bb04ccf-geoclue.service-Dsk9VR/»: Отказано в доступе
warning: не удалось открыть каталог «systemd-private-880b4b7bc6f244d1a9b341b39bb04ccf-colord.service-oRFOR0/»: Отказано в доступе
warning: не удалось открыть каталог «systemd-private-880b4b7bc6f244d1a9b341b39bb04ccf-ModemManager.service-Sl8pAY/»: Отказано в доступе
warning: не удалось открыть каталог «systemd-private-880b4b7bc6f244d1a9b341b39bb04ccf-dbus-broker.service-jzhT53/»: Отказано в доступе
warning: не удалось открыть каталог «systemd-private-880b4b7bc6f244d1a9b341b39bb04ccf-chronyd.service-TDILZ5/»: Отказано в доступе
warning: не удалось открыть каталог «systemd-private-880b4b7bc6f244d1a9b341b39bb04ccf-upower.service-TN7Aqq/»: Отказано в доступе
warning: не удалось открыть каталог «systemd-private-880b4b7bc6f244d1a9b341b39bb04ccf-systemd-logind.service-Fzpe0D/»: Отказано в доступе
warning: не удалось открыть каталог «systemd-private-880b4b7bc6f244d1a9b341b39bb04ccf-switcheroo-control.service-ddlw8r/»: Отказано в доступе
warning: не удалось открыть каталог «systemd-private-880b4b7bc6f244d1a9b341b39bb04ccf-rtkit-daemon.service-0s3h1Q/»: Отказано в доступе
warning: не удалось открыть каталог «systemd-private-880b4b7bc6f244d1a9b341b39bb04ccf-power-profiles-daemon.service-0bhuDM/»: Отказано в доступе
warning: не удалось открыть каталог «systemd-private-880b4b7bc6f244d1a9b341b39bb04ccf-low-memory-monitor.service-iHSbIH/»: Отказано в доступе
warning: не удалось открыть каталог «systemd-private-880b4b7bc6f244d1a9b341b39bb04ccf-systemd-resolved.service-YgIdJq/»: Отказано в доступе
warning: не удалось открыть каталог «systemd-private-880b4b7bc6f244d1a9b341b39bb04ccf-systemd-oomd.service-0XRixB/»: Отказано в доступе
warning: добавление встроенного git репозитория: gitflow
подсказка: You've added another git repository inside your current repository.
подсказка: Clones of the outer repository will not contain the contents of
подсказка: the embedded repository and will not know how to obtain it.
подсказка: If you meant to add a submodule, use:
подсказка:
подсказка:     git submodule add <url> gitflow
подсказка:
подсказка: If you added this path by mistake, you can remove it from the
подсказка: index with:
подсказка:
подсказка:     git rm --cached gitflow
подсказка:
подсказка: See "git help submodule" for more information.
[adzayjceva@adzayjceva tmp]$ git commit -am 'chore(main): add version'
[release/1.0.0 54af9d2] chore(main): add version
13 files changed, 123 insertions(+)
create mode 100644 .X0-lock
create mode 100644 .X1-lock
create mode 100644 .X1024-lock
create mode 100644 .X1025-lock
create mode 100644 Temp-611d644b-ea08-4f49-8c8e-c484aa54f4f5/mesa_shader_cache/1a/b708827ed73defff0c5a582eed96534c6ddc9b
create mode 100644 Temp-611d644b-ea08-4f49-8c8e-c484aa54f4f5/mesa_shader_cache/5b/22097763442e7dcf7f2867377b7314c03ad8e6
create mode 100644 Temp-611d644b-ea08-4f49-8c8e-c484aa54f4f5/mesa_shader_cache/cc/35802c8028040f33c11a75141f2cf8100cc755
create mode 100644 Temp-611d644b-ea08-4f49-8c8e-c484aa54f4f5/mesa_shader_cache/index
create mode 100644 VERSION
create mode 100000 gitflow
create mode 100755 gitflow-installer.sh
create mode 100644 lu4644c4j7tq.tmp/lu4644c4j7ts.tmp
create mode 100644 mozilla_adzayjceva0/Lab01_summary.docx
```

Рис. 18

6. Зальём релизную ветку в основную ветку (Рис. [-@fig:019]):

```
[adzayjceva@adzayjceva tmp]$ get flow release finish 1.0.0
bash: get: command not found...
Similar command is: 'git'
[adzayjceva@adzayjceva tmp]$ git flow release finish 1.0.0
warning: unable to unlink '.X1024-lock': Операция не позволена
warning: unable to unlink '.X1025-lock': Операция не позволена
warning: unable to rmdir 'gitflow': Операция не позволена
Переключено на ветку «master»
error: Указанные неотслеживаемые файлы в рабочем каталоге будут перезаписаны при слиянии:
.X1024-lock
.X1025-lock
Переместите эти файлы или удалите их перед переключением веток.
Прерываю
Fatal: There were merge conflicts.
[adzayjceva@adzayjceva tmp]$
```

Рис. 19

7. Отправим данные на github:

*git push —all git push —tags* Возникла ошибка. На этом лабораторная работа завершена.

## Контрольные вопросы:

1. Система контроля версий Git представляет собой набор программ командной строки. Доступ к ним можно получить из терминала посредством ввода команды git с различными опциями. Системы контроля версий (Version Control System, VCS) применяются при работе нескольких человек над одним проектом.
2. В классических системах контроля версий используется централизованная модель, предполагающая наличие единого репозитория для хранения файлов. Выполнение большинства функций по управлению версиями осуществляется специальным сервером. Участник проекта (пользователь) перед началом работы посредством определённых команд получает нужную ему версию файлов. После внесения изменений, пользователь размещает новую версию в хранилище. При этом предыдущие версии не удаляются из центрального хранилища и к ним можно вернуться в любой момент. Сервер может сохранять неполную версию изменённых файлов, а производить так называемую дельта-компрессию — сохранять только изменения между последовательными версиями, что позволяет уменьшить объём хранимых данных. Системы контроля версий также могут обеспечивать дополнительные, более гибкие функциональные возможности. Например, они могут поддерживать работу с несколькими версиями одного файла, сохраняя общую историю изменений до точки ветвления версий и собственные истории изменений каждой ветви. Кроме того, обычно доступна информация о том, кто из участников, когда и какие изменения вносил. Обычно такого рода информация хранится в журнале изменений, доступ к которому можно ограничить.
3. Централизованные системы — это системы, которые используют архитектуру клиент / сервер, где один или несколько клиентских узлов напрямую подключены к центральному серверу. Пример - Wikipedia. В децентрализованных системах каждый узел принимает свое собственное решение. Конечное поведение системы является совокупностью решений отдельных узлов. Пример

— Bitcoin. В классических системах контроля версий используется централизованная модель, предполагающая наличие единого репозитория для хранения файлов. Выполнение большинства функций по управлению версиями осуществляется специальным сервером.

4. Создадим локальный репозиторий. Сначала сделаем предварительную конфигурацию, указав имя и email владельца репозитория: `git config --global user.name "Имя Фамилия"` `git config --global user.email "work@mail"` и настроив utf-8 в выводе сообщений git: `git config --global core.quotePath false` Для инициализации локального репозитория, расположенного, например, в каталоге `~/tutorial`, необходимо ввести в командной строке: `cd mkdir tutorial cd tutorial git init`
5. Для последующей идентификации пользователя на сервере репозитория необходимо сгенерировать пару ключей (приватный и открытый): `ssh-keygen -C "Имя Фамилия work@mail"` Ключи хранятся в каталоге `~/.ssh/`. Скопировав из локальной консоли ключ в буфер обмена `cat ~/.ssh/id_rsa.pub | xclip -sel clip` вставляем ключ в появившееся на сайте поле.
6. У Git две основных задачи: первая — хранить информацию о всех изменениях в вашем коде, начиная с самой первой строчки, а вторая — обеспечение удобства командной работы над кодом.
7. Основные команды git: **Наиболее часто используемые команды git:** – создание основного дерева репозитория: `git init` – получение обновлений (изменений) текущего дерева из центрального репозитория: `git pull` – отправка всех произведённых изменений локального дерева в центральный репозиторий: `git push` – просмотр списка изменённых файлов в текущей директории: `git status` – просмотр текущих изменений: `git diff` – сохранение текущих изменений: – добавить все изменённые и/или созданные файлы и/или каталоги: `git add .` – добавить конкретные изменённые и/или созданные файлы и/или каталоги: `git add имена_файлов` – удалить файл и/или каталог из индекса репозитория (при этом файл и/или каталог остаётся в локальной директории): `git rm имена_файлов` – сохранение добавленных изменений: – сохранить все добавленные изменения и все изменённые файлы: `git commit -am 'Описание коммита'` – сохранить добавленные изменения с внесением комментария через встроенный редактор: `git commit` – создание новой ветки, базирующейся на текущей: `git checkout -b имя_ветки` – переключение на некоторую ветку: `git checkout имя_ветки` (при переключении на ветку, которой ещё нет в локальном репозитории, она будет создана и связана с удалённой) – отправка изменений конкретной ветки в центральный репозиторий: `git push origin имя_ветки` – слияние ветки стекущим деревом: `git merge --no-ff имя_ветки` – удаление ветки: – удаление локальной уже слитой с основным деревом ветки: `git branch -d имя_ветки` – принудительное удаление локальной ветки: `git branch -D имя_ветки` – удаление ветки с центрального репозитория: `git push origin :имя_ветки`
8. Использование git при работе с локальными репозиториями (добавления текстового документа в локальный репозиторий): `git add hello.txt git commit -am 'Новый файл'`

9. Проблемы, которые решают ветки git:

- нужно постоянно создавать архивы с рабочим кодом
- сложно “переключаться” между архивами
- сложно перетаскивать изменения между архивами
- легко что-то напутать или потерять

10. Во время работы над проектом так или иначе могут создаваться файлы, которые не требуется добавлять в последствии в репозиторий. Например, временные файлы, создаваемые редакторами, или объектные файлы, создаваемые компиляторами. Можно прописать шаблоны игнорируемых при добавлении в репозиторий типов файлов в файл.gitignore с помощью сервисов. Для этого сначала нужно получить списки меняющихся шаблонов: `curl -L -s https://www.gitignore.io/api/list` Затем скачать шаблон, например, для С и С++ `curl -L -s https://www.gitignore.io/api/c >> .gitignore` `curl -L -s https://www.gitignore.io/api/c++ >> .gitignore`

## Вывод

Я изучила идеологию и применение контроля версий.