**РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ**

**Факультет физико-математических и естественных наук**

**Кафедра прикладной информатики и теории вероятностей**

**ОТЧЁТ**

**ПО ЛАБОРАТОРНОЙ РАБОТЕ №2**

*дисциплина: Архитектура компьютера*

Студент:

Гольцова Мария

Группа:

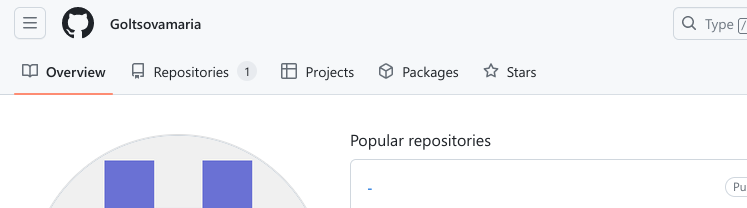
НММбд-01-23

**МОСКВА**

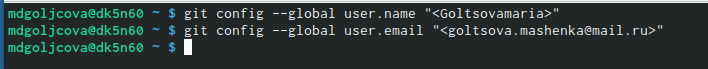
**2023 г.**

**Цель работы:** изучить идеологию и применения средств контроля версий, а также приобрести практические навыки по работе с системой git.

**Ход работы**

Рис. 1

Создала учётную запись на сайте <https://github.com/> и заполнила основные данные.

Рис. 2

Сначала сделала предварительную конфигурацию git. Открыла терминал и ввела следующие команды, указав имя и email владельца репозитория.

Рис. 3

Настроила utf-8 в выводе сообщений git.

Рис. 4

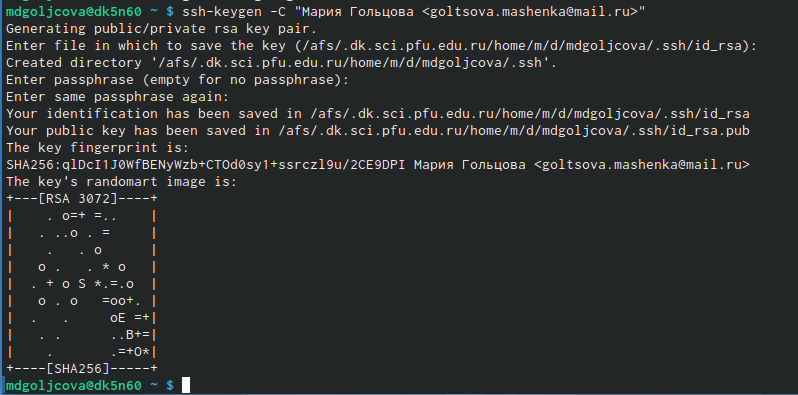
Задала имя начальной ветки (master).

Рис. 5

Параметр autocrlf.

Рис. 6

Параметр safecrlf.

Рис. 7

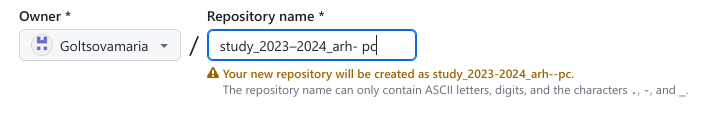
Для последующей идентификации пользователя на сервере репозиториев сгенерировала пару ключей (приватный и открытый).

Рис. 8

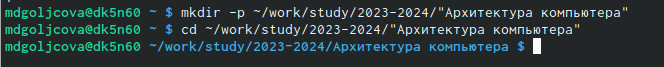
Зашла на сайт http: //github.org/ под своей учётной записью и перешла в меню. После этого выбрала в боковом меню SSH and GPG keys и нажала кнопку New SSH key. Скопировав из локальной консоли ключ в буфер обмена, вставила ключ в появившееся на сайте поле и указала для ключа имя (Title).

Рис. 9

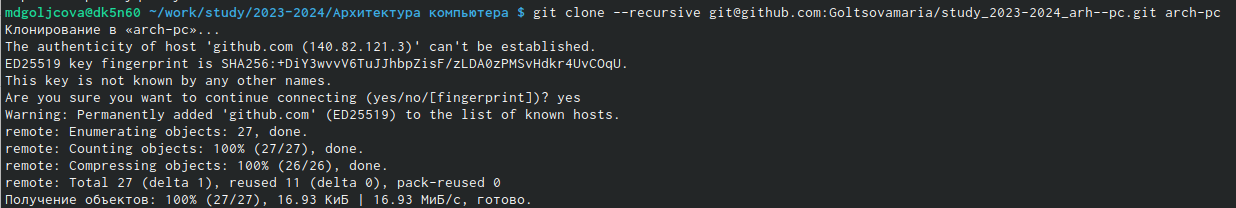
Открыла терминал и создала каталог для предмета «Архитектура компьютера».

Рис. 10

Перешла на страницу репозитория с шаблоном курса https://github.com/yamadharma/cour se-directory-student-template. Далее выбрала Use this template. В открывшемся окне задала имя репозитория (Repository name) study\_2023–2024\_arhpc и создала репозиторий.

Рис. 11

Открыла терминал и перешла в каталог курса.

Рис. 12

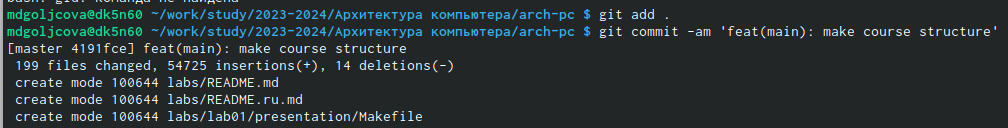
Клонировала созданный репозиторий.

Рис. 13

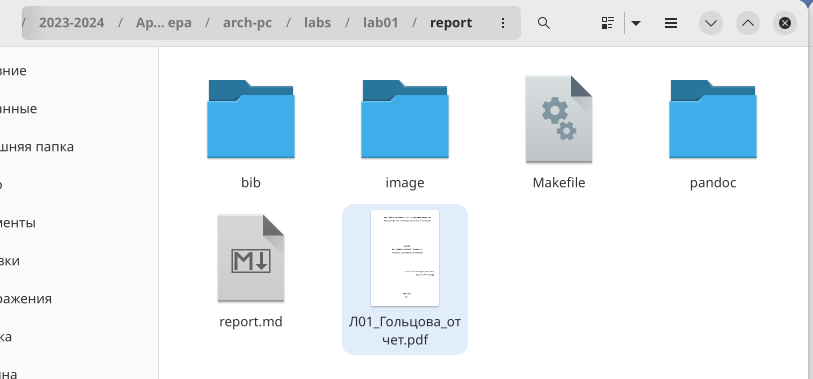
Перешла в каталог курса и удалила лишние файлы.

Рис. 14

Создала необходимые каталоги.

Рис. 15

Отправила файлы на сервер.

Рис. 15

Скопировала отчёты по выполнению лабораторной работы №1 в соответствующий каталог созданного рабочего пространства.

**Контрольные вопросы для самопроверки.**

1. Система контроля версий — это программное обеспечение, позволяющее управлять изменениями. Системы контроля версий (Version Control System, VCS) применяются при работе нескольких человек над одним проектом. Можно объединить (слить) изменения, сделанные разными участниками (автоматически или вручную), вручную выбрать нужную версию, отменить изменения вовсе или заблокировать файлы для изменения. Системы контроля версий могут поддерживать работу с несколькими версиями одного файла, сохраняя общую историю изменений до точки ветвлений версий и собственные истории изменений каждой ветви.
2. Репозиторий (repository) — специальное хранилище файлов и папок проекта, изменения которых отслеживаются. В распоряжении разработчика имеется «рабочая копия» (working copy) проекта, с которой он непосредственно работает. Рабочую копию необходимо периодически синхронизировать с репозиторием, эта операция предполагает отправку в него изменений, которые пользователь внёс в свою рабочую копию (такая операция называется commit) и актуализацию рабочей копии, в процессе которой к пользователю загружается последняя версия из репозитория (этот процесс назвается update).
3. При использовании централизованной архитектуры VCS представляет из себя серверное приложение. Центральный сервер непосредственно хранит файлы проекта и их историю, а клиентские приложения запрашивают у сервера необходимые им версии интересующих их файлов; все пользователи подключены к центральному владельцу сети или сервера. Недостаток -необходимость наличия доступа к сети при совершении операций с репозиторием. Децентрализованные VCS не требуют доступа к серверу при совершении операций с репозиторием; нет единого центрального владельца. Недостатки DVCS — большой размер рабочей копии, сложности при обмене изменениями с другими участниками. Пример централизованной системы — CVS. Пример децентрализованной системы — Интернет.
4. Работа пользователя со своей веткой начинается с проверки и получения изменений из центрального репозитория. Затем можно вносить изменения в локальном дереве и (или) ветке. После завершения внесения какого-то изменения в файлы и (или) каталоги проекта необходимо разместить их в центральном репозитории.
5. Обновить проект, забрать последнюю версию из репозитория, внести изменения в репозиторий, разрешить конфликты, создать бранч (ветку).
6. Git позволяет сохранять контрольные точки проекта и помогает устранять конфликты.
7. git init — создание основного дерева репозитория; git pull — получение обновлений; git push — отправка изменений в центральный репозиторий; git status — просмотр списка изменённых файлов; git diff — просмотр текущих изменений; git add - добавить все изменённые и/или созданные файлы и/или каталоги; git rm — удалить файл и/или каталог из индекса репозитория; git commit -am - сохранить все добавленные изменения и все изменённые файлы; git checkout -b - создание новой ветки, базирующейся на текущей; git checkout - переключение на некоторую ветку; git push origin - отправка изменений конкретной ветки в центральный репозиторий; git merge —no--ff - слияние ветки с текущим деревом; git branch -d - удаление локальной уже слитой с основным деревом ветки; git branch -D - принудительное удаление локальной ветки.
8. Локальный репозиторий расположен непосредственно в памяти компьютера разработчика, в нём происходит разработка и фиксаця изменений, после чего можно отправить на удалённый репозиторий. Удалённый репозиторий расположен на сервере, может быть приватным — доступным ограниченному числу лиц, и публичным.

**Вывод:** я изучила идеологию и применение средств контроля версий, а также приобрела практические навыки по работе с системой git.