Отчёт по лабораторной работе №1

Математическое моделирование

Самсонова Мария Ильинична

Содержание

[Цель работы 1](#_Toc158417697)

[Задание 1](#_Toc158417698)

[Теоретическое введение 1](#_Toc158417699)

[Выполнение лабораторной работы 2](#_Toc158417700)

[Вывод 9](#_Toc158417701)

# Цель работы

Подробное изучение идеологии и применение различных средств контроля версий, освоение практических навыков по работе с git.

# Задание

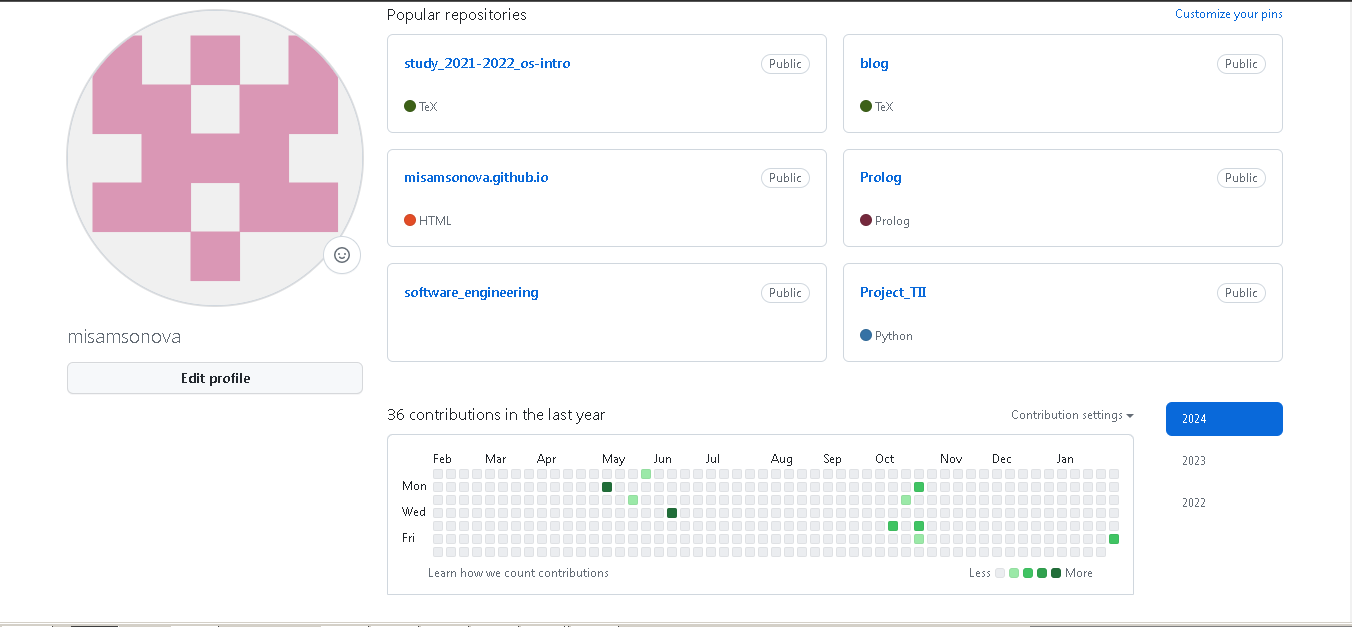
* Создать базовую конфигурацию для работы с git.
* Создать ключ SSH.
* Создать ключ PGP.
* Настроить подписи git.
* Зарегистрироваться на Github.
* Создать локальный каталог для выполнения заданий по предмету.

# Теоретическое введение

Системы контроля версий (Version Control System, VCS) применяются при работе нескольких человек над одним проектом. Обычно основное дерево проекта хранится в локальном или удалённом репозитории, к которому настроен доступ для участников проекта. При внесении изменений в содержание проекта система контроля версий позволяет их фиксировать, совмещать изменения, произведённые разными участниками проекта, производить откат к любой более ранней версии проекта, если это требуется. 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 не полную версию изменённых файлов, а производить так называемую дельтакомпрессию — сохранять только изменения между последовательными версиями, что позволяет уменьшить объём хранимых данных. Системы контроля версий поддерживают возможность отслеживания и разрешения конфликтов, которые могут возникнуть при работе нескольких человек над одним файлом. Можно объединить (слить) изменения, сделанные разными участниками (автоматически или вручную), вручную выбрать нужную версию, отменить изменения вовсе или заблокировать файлы для изменения. В зависимости от настроек блокировка не позволяет другим пользователям получить рабочую копию или препятствует изменению рабочей копии файла средствами файловой системы ОС, обеспечивая таким образом, привилегированный доступ только одному пользователю, работающему с файлом. 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. В отличие от классических, в распределённых системах контроля версий центральный репозиторий не является обязательным. Среди классических VCS наиболее известны CVS, Subversion, а среди распределённых — Git, Bazaar, Mercurial. Принципы их работы схожи, отличаются они в основном синтаксисом используемых в работе команд.

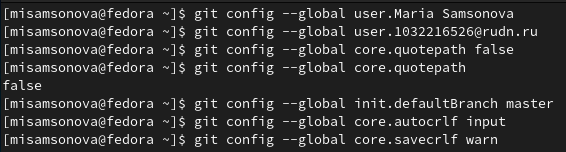
# Выполнение лабораторной работы

1. Открываем свою учётную запись на https://github.com: (рис. [-@fig:001])



Учетная запись на github

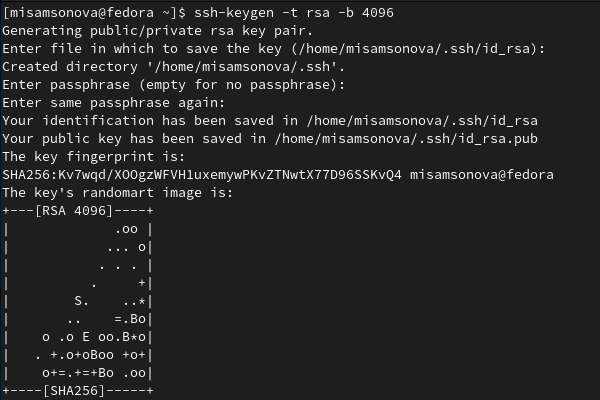
1. Задаём имя и email владельца репозитория данными командами git config –global user.name “Name Surname” и git config –global user.email “work@mail”, настраиваем utf-8 в выводе сообщений git командой git config –global core.quotepath false, а также задаём имя начальной ветки (будем называть её master): (рис. [-@fig:002])



Задание имени и email пользователя, настройка utf-8 в выводе сообщений, задание имени начальной ветки

{#fig:002 width=70% }

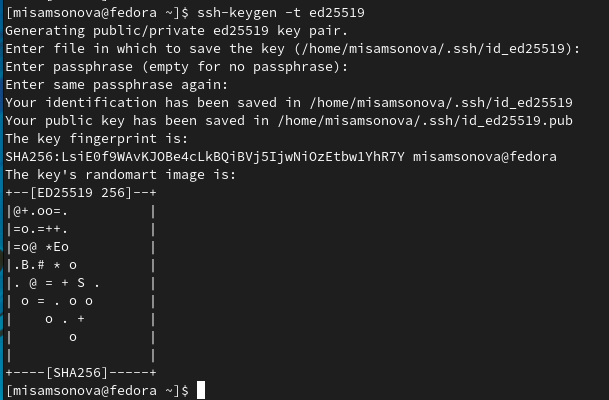
1. Создаём ключей ssh по алгоритму rsa размером 4096 бит и по алгоритму ed25519: (рис. [-@fig:003])



Создание ключа ssh по алгоритму rsa с ключём размером 4096 бит

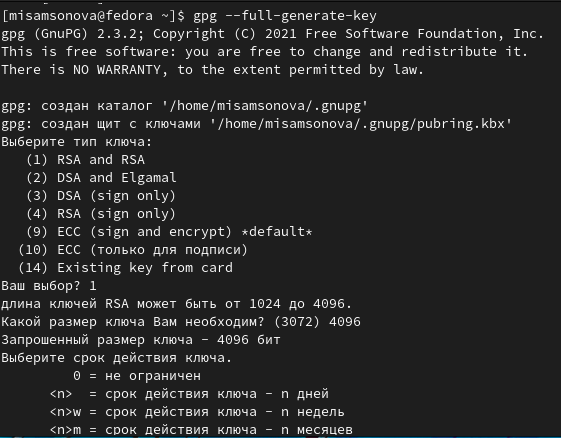
{#fig:003 width=70% }

(рис. [-@fig:004])



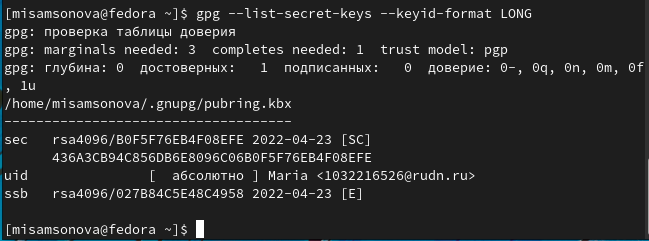
Создание ключа ssh по алгоритму ed25519

{#fig:004 width=70% }

1. Создаём ключ GPG, генерируя его и определяя различные характеристики: (рис. [-@fig:005]) 

{#fig:005 width=70% }

1. Добавляем GPG ключ в GitHub, выводя список ключей и копируя отпечаток приватного ключа: (рис.[-@fig:006])



Данные о ключе GPG

{#fig:006 width=70% }

1. Получаем содержимое ключа GPG и скопируем наш сгенерированный ключ в буфер обмена: (рис. [-@fig:007]) и (рис. [-@fig:008])



Данные о ключе GPG

{#fig:007 width=70% }



Данные о ключе GPG

{#fig:008 width=70% }

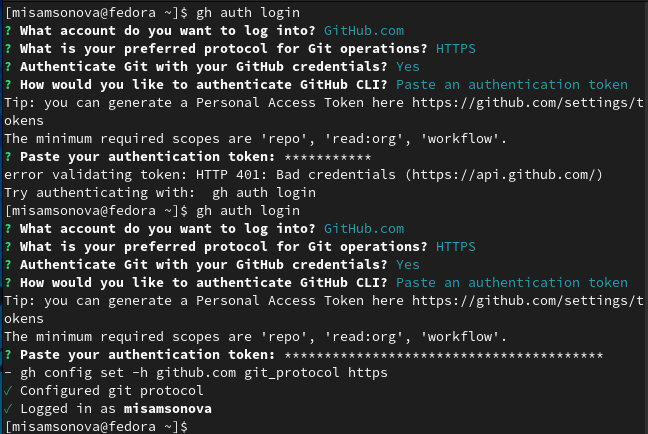
1. Переходим в настройки GitHub, нажимаем на кнопку New GPG key и вставляем полученный ключ в поле ввода. Получаем: (рис. [-@fig:009])

Настройка подписей коммитов git

Настройка подписей коммитов git

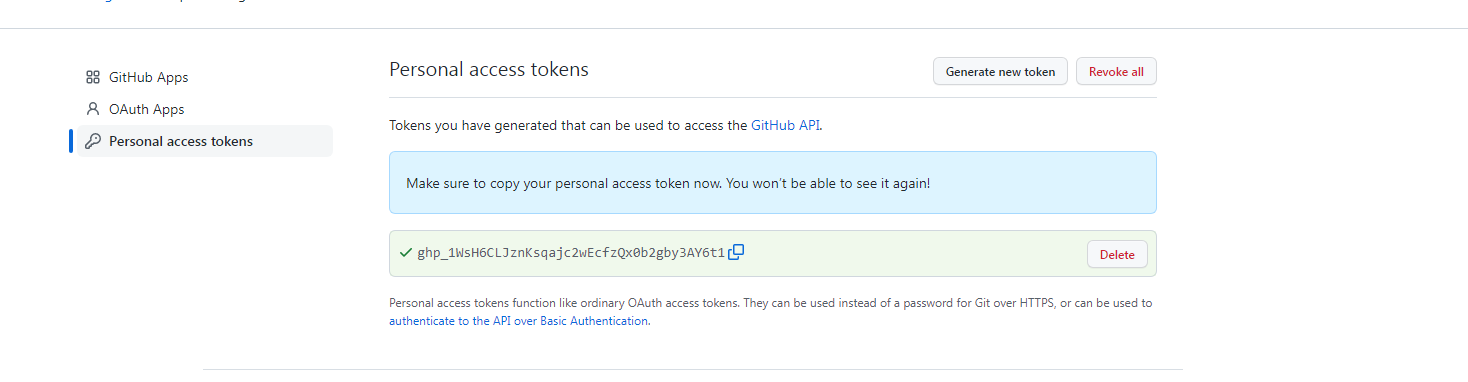
{#fig:009 width=70% }

1. Настраиваем gh и для начала авторизуемся в github. Далее создаём персональный токен, чтобы не нужно было входить в свою учётную запись через веб-сайт: (рис. [-@fig:010]) и (рис. [-@fig:011])



Вход в учётную запись github

{#fig:010 width=70% }



Создание токена для авторизации

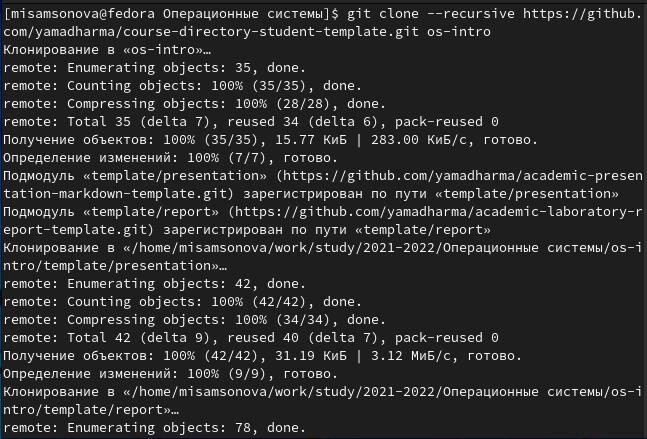
{#fig:011 width=70% }

1. После чего создаём репозиторий курса на основе шаблона следующим образом:(рис. [-@fig:012]) и (рис. [-@fig:013])

Создание репозитория курса на основе шаблона

Создание репозитория курса на основе шаблона

{#fig:012 width=70% }



Создание репозитория курса на основе шаблона

{#fig:013 width=70% }

1. Далее настраиваем каталога курса, то есть переходим в каталог курса командой cd ~/work/study/2023-2024/“Математическое моделирование”/mathmod и следующим образом удаляем лишние файлы, создаём необходимые каталоги и отправляем файлы на сервер: (рис. [-@fig:014])

Настройка каталога курса

Настройка каталога курса

{#fig:014 width=70% }

# Вывод

В процессе выполнения данной лабораторной работы мы приобрели практические навыки применения средств контроля версий и вспомнили, как работать с git.