

# **Лабораторная работа №2**

**Операционные системы**

Клименко Алёна Сергеевна

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>5</b>
<b>2</b>	<b>Задание</b>	<b>6</b>
<b>3</b>	<b>Теоретическое введение</b>	<b>7</b>
<b>4</b>	<b>Выполнение лабораторной работы</b>	<b>9</b>
4.1	Установка git и gh . . . . .	9
4.2	Базовая настройка git. . . . .	9
4.3	Создание ssh ключа. . . . .	10
4.3.1	Добавление gpg-ключа в учетную запись ГитХаb. . . . .	10
4.4	Создание PGP ключа. . . . .	11
4.4.1	Добавление ключа на ГитХаb. . . . .	12
4.4.2	Настройка автоматических подписей коммитов git . . . . .	12
4.5	Создание и настройка репозитория курса. . . . .	13
<b>5</b>	<b>Контрольные вопросы</b>	<b>14</b>
<b>6</b>	<b>Выводы</b>	<b>18</b>
	<b>Список литературы</b>	<b>19</b>

# Список иллюстраций

4.1	Ввод команд в терминал . . . . .	9
4.2	Создание ключа . . . . .	10
4.3	Новый ключ ssh . . . . .	11
4.4	Создание шаблона . . . . .	11
4.5	гит . . . . .	12
4.6	Новый ключ PGP . . . . .	12
4.7	Настройка необходимых подписей коммитов . . . . .	13
4.8	git . . . . .	13

## **Список таблиц**

# 1 Цель работы

Целью данной работы является изучение идеологии и применения средств контроля версий и освоение умения по работе с git.

## 2 Задание

1. Создать базовую конфигурацию для работы с git.
2. Создать ключ SSH.
3. Создать ключ PGP.
4. Настроить подписи git.
5. Зарегистрироваться на Github.
6. Создать локальный каталог для выполнения заданий по предмету.

### 3 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 дельта-компрессию — 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.



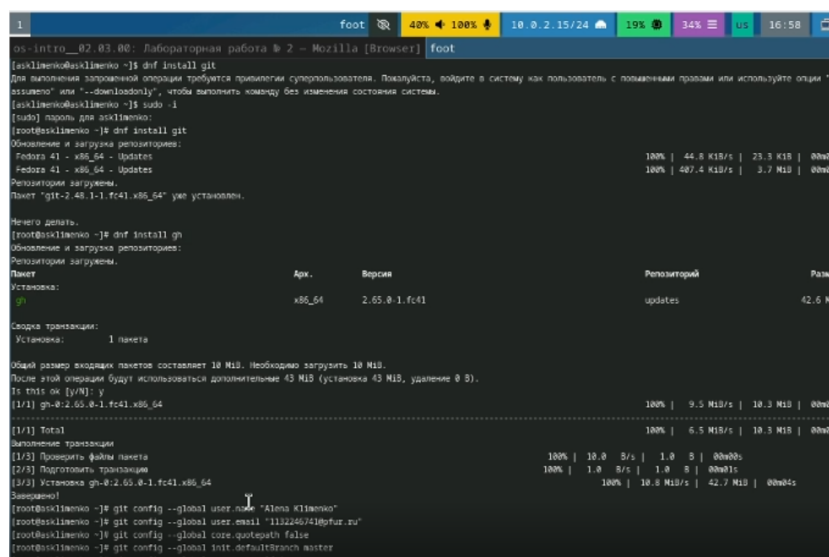
## 4 Выполнение лабораторной работы

### 4.1 Установка git и gh

Установим гит командой `dnf install git`, установим gh командой `dnf install gh`

### 4.2 Базовая настройка git.

Открываем терминал. При помощи команд `git config --global user.name` и `git config --global user.email` зададим имя пользователя и адрес электронной почты. При помощи команды `git config --global core.quotepath false` настроим utf-8 в выводе сообщений git. При помощи команды `git config --global init.defaultBranch master` зададим начальной ветке имя master. (рис. 4.1)



```
1
foot 40% 100% 18.0.2.15/24 19% 34% 16:58
os-intro_02.03.00: Лабораторная работа № 2 – Mozilla [Browser] foot
[askillenko@askillenko ~]$ dnf install git
Для выполнения запрошенной операции требуется привилегия суперпользователя. Пожалуйста, войдите в систему как пользователь с полными правами или используйте опцию "--assumeno" или "--downloadonly", чтобы выполнить команду без изменения состояния системы.
[askillenko@askillenko ~]$ sudo -i
[sudo] пароль для askillenko:
[root@askillenko ~]# dnf install git
Обновление и загрузка репозитория:
Fedora 41 - x86_64 - Updates 100% | 44.8 KiB/s | 23.3 KiB | 00h01s
Fedora 41 - x86_64 - Updates 100% | 487.4 KiB/s | 3.7 MiB | 00h00s
Репозитории загружены.
Пакет "git-2.45.1-1.fc41.x86_64" уже установлен.

Ничего делать.
[root@askillenko ~]# dnf install gh
Обновление и загрузка репозитория:
Репозитории загружены.
Пакет: Арх. Версия Репозиторий Размер
Установка: gh x86_64 2.65.0-1.fc41 updates 42.6 MiB

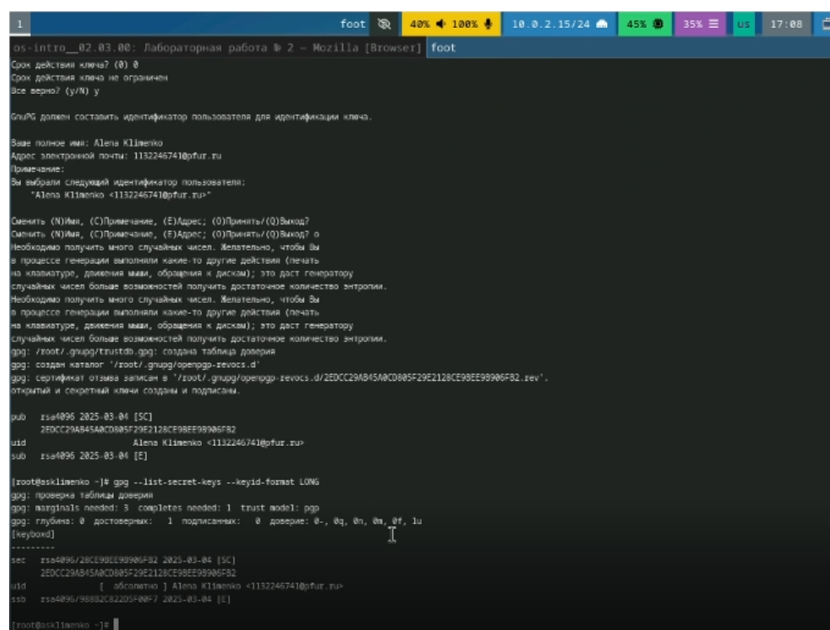
Сводка транзакции:
Установка: 1 пакета

Общий размер входящих пакетов составляет 10 MiB. Необходимо загрузить 10 MiB.
После этой операции будут использоваться дополнительные 43 MiB (установка 43 MiB, удаление 0 B).
It will be [Y/n] y
(1/1) gh-2.65.0-1.fc41.x86_64 100% | 9.5 MiB/s | 10.3 MiB | 00h01s
(1/1) Total 100% | 6.5 MiB/s | 10.3 MiB | 00h02s
Выполнение транзакции
(1/3) Проверить файлы пакета 100% | 10.0 B/s | 1.0 B | 00h00s
(2/3) Подготовить транзакцию 100% | 1.0 B/s | 1.0 B | 00h01s
(3/3) Установка gh-2.65.0-1.fc41.x86_64 100% | 10.8 MiB/s | 42.7 MiB | 00h04s
Завершено!
[root@askillenko ~]# git config --global user.name "Aleks Krasenko"
[root@askillenko ~]# git config --global user.email "1112246743@yur.ru"
[root@askillenko ~]# git config --global core.quotepath false
[root@askillenko ~]# git config --global init.defaultBranch master
[root@askillenko ~]# git config --global core.autocrlf input
```

Рис. 4.1: Ввод команд в терминал

## 4.3 Создание ssh ключа.

создания ключа (рис. 4.2)



```
1
os-intro_02.03.00: Лабораторная работа № 2 - Mozilla [Browser] foot
Срок действия ключа? (0) 0
Срок действия ключа не ограничен
Все верно? (y/N) y

ssh-keygen должен составить идентификатор пользователя для идентификации ключа.

Ваше полное имя: Alena Klimenko
Адрес электронной почты: 1132246741@pfur.ru
Примечание:
Вы выбрали следующий идентификатор пользователя:
    "Alena Klimenko <1132246741@pfur.ru>"

Создать (Y/n), (C)Примечание, (E)Адрес, (O)Применить/(O)Вывод?
Создать (Y/n), (C)Примечание, (E)Адрес, (O)Применить/(O)Вывод? 0
Необходимо получить много случайных чисел. Желательно, чтобы вы
в процессе генерации выполняли какие-то другие действия (печатать
на клавиатуре, двигать мышью, обращаться к диску); это даст генератору
случайных чисел больше возможностей получить достаточное количество энтропии.
Необходимо получить много случайных чисел. Желательно, чтобы вы
в процессе генерации выполняли какие-то другие действия (печатать
на клавиатуре, двигать мышью, обращаться к диску); это даст генератору
случайных чисел больше возможностей получить достаточное количество энтропии.
gpg: /root/.gnupg/trustdb.gpg: создана таблица доверия
gpg: создан каталог '/root/.gnupg/openpgp-revocs.d'
gpg: сертификат отменяется записан в '/root/.gnupg/openpgp-revocs.d/2EDCC29A845A8C9A85F29E2128CE98EE9898F52.rev'.
Открытый и секретный ключи созданы и подписаны.

pub rsa4096 2025-03-04 [SC]
    2EDCC29A845A8C9A85F29E2128CE98EE9898F52
uid          Alena Klimenko <1132246741@pfur.ru>
sub rsa4096 2025-03-04 [S]

[root@osklimenko ~]# gpg --list-secret-keys --keyid-format LONG
gpg: проверка таблицы доверия
gpg: marginals needed: 3 completes needed: 1 trust model: gpg
gpg: ryubms: 0 достоверных: 1 подписанных: 0 доверие: 0-, 0q, 0n, 0f, 1u
(Keyboard)
-----
sec rsa4096/2EDCC29A845A8C9A85F29E2128CE98EE9898F52 2025-03-04 [SC]
    2EDCC29A845A8C9A85F29E2128CE98EE9898F52
uid          [ абсолютное ] Alena Klimenko <1132246741@pfur.ru>
ssb rsa4096/9885C2A220F48077 2025-03-04 [S]

[root@osklimenko ~]#
```

Рис. 4.2: Создание ключа

### 4.3.1 Добавление gpg-ключа в учетную запись ГитХаба.

Копируем созданный ключ и переносим его на сайт гитхаб в раздел ssh и gpg keys.

Создаем новый ключ, задаем ему название и переносим ключ в поле key, добавляем ключ на сайт. (рис. 4.3)

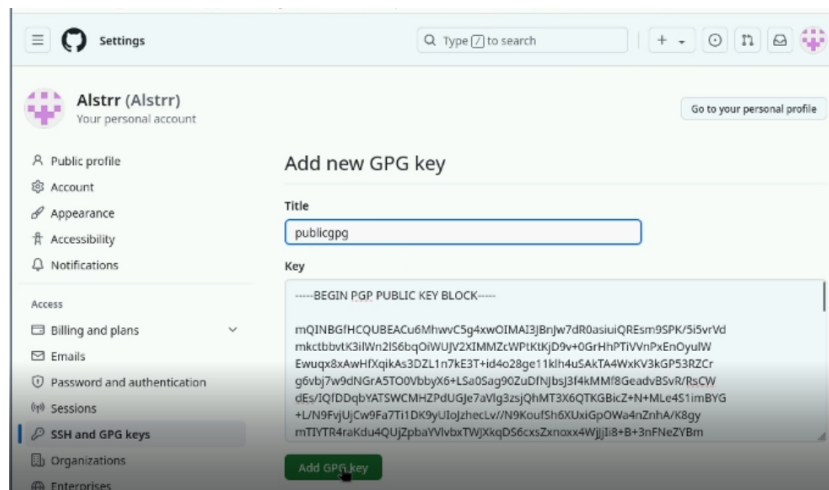


Рис. 4.3: Новый ключ ssh

## 4.4 Создание PGP ключа.

создаем шаблон рабочего пространства (рис. 4.4)

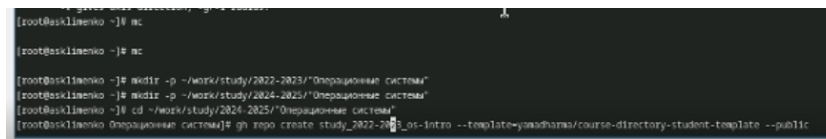


Рис. 4.4: Создание шаблона

связываем аккаунт гит с виртуальной машиной (рис. 4.5)

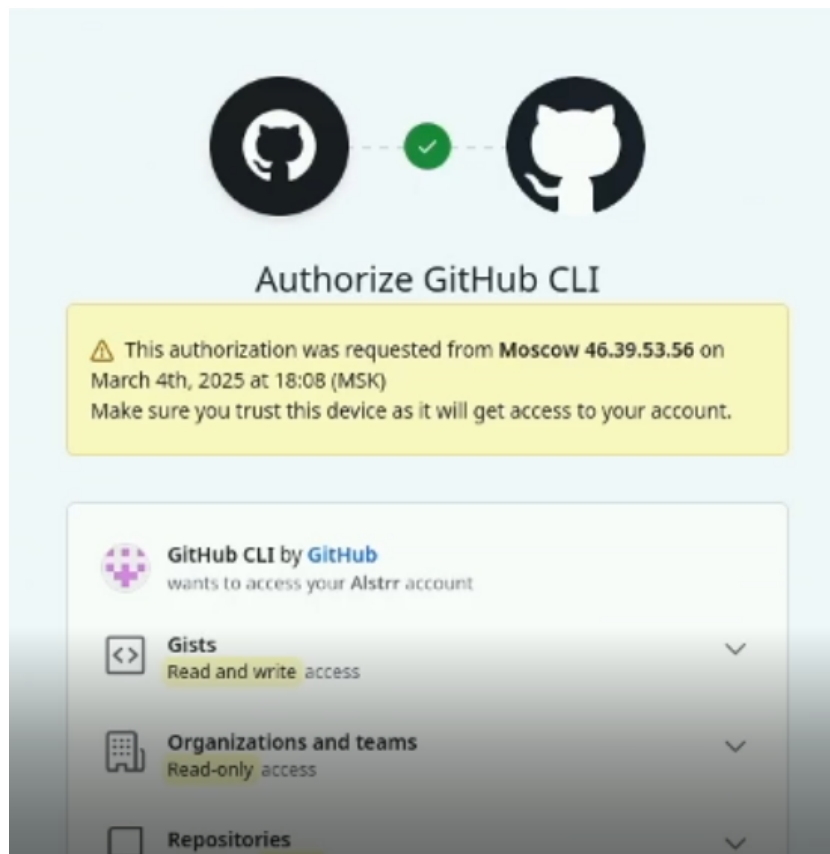


Рис. 4.5: гит

#### 4.4.1 Добавление ключа на ГитХаб.

создаем папки для лабораторных работ (рис. 4.6)

```
[root@asklimenko ~]# cd ~/work/study/2024-2025/Операционные системы/os-intro/
[root@asklimenko os-intro]# ls
CHANGELOG.md  config  COURSE  LICENSE  Makefile  package.json  README.en.md  README.git-flow.md  README.md  template
[root@asklimenko os-intro]# rm package.json
rm: удалить обычный файл 'package.json'? y
[root@asklimenko os-intro]# echo os-intro > COURSE
[root@asklimenko os-intro]#
```

Рис. 4.6: Новый ключ PGP

#### 4.4.2 Настройка автоматических подписей коммитов git

выгружаем информацию в git (рис. 4.7)

```

[root@asklimenko os-intro]# git add .
[root@asklimenko os-intro]# git commit -am 'feat(main): make course structure'
[master 50399c4] feat(main): make course structure
2 files changed, 1 insertion(+), 14 deletions(-)
delete mode 100644 package.json
[root@asklimenko os-intro]# git push

```

Рис. 4.7: Настройка необходимых подписей коммитов

## 4.5 Создание и настройка репозитория курса.

Отправляем файлы первой лабораторной работы на сервер.

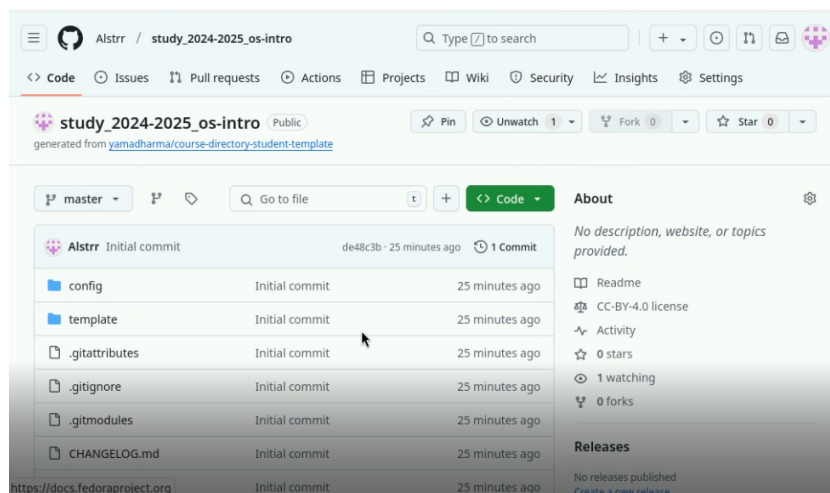


Рис. 4.8: git

## 5 Контрольные вопросы

1. Что такое системы контроля версий (VCS) и для решения каких задач они предназначаются?

Система контроля версий — программное обеспечение для облегчения работы с изменяющейся информацией. Система управления версиями позволяет хранить несколько версий одного и того же документа, при необходимости возвращаться к более ранним версиям, определять, кто и когда сделал то или иное изменение, и многое другое. Системы контроля версий (Version Control System, VCS) применяются для:

- Хранение полной истории изменений причин всех производимых изменений
- Откат изменений, если что-то пошло не так
- Поиск причины и ответственного за появления ошибок в программе
- Совместная работа группы над одним проектом
- Возможность изменять код, не мешая работе других пользователей

2. Объясните следующие понятия VCS и их отношения: хранилище, commit, история, рабочая копия

Репозиторий - хранилище версий - в нем хранятся все документы вместе с историей их изменения и другой служебной информацией. Commit — отслеживание изменений Рабочая копия - копия проекта, связанная с репозиторием (текущее состояние файлов проекта, основанное на версии из хранилища (обычно на последней) История хранит все изменения в проекте и позволяет при необходимости обратиться к нужным данным.

3. Что представляют собой и чем отличаются централизованные и децентрализованные VCS? Приведите примеры VCS каждого вида

Централизованные VCS (Subversion; CVS; TFS; VAULT; AccuRev): Одно основное хранилище всего проекта Каждый пользователь копирует себе необходимые ему файлы из этого репозитория, изменяет и, затем, добавляет свои изменения обратно Децентрализованные VCS (Git; Mercurial; Bazaar): У каждого пользователя свой вариант (возможно не один) репозитория Присутствует возможность добавлять и забирать изменения из любого репозитория . В классических системах контроля версий используется централизованная модель, предполагающая наличие единого репозитория для хранения файлов. Выполнение большинства функций по управлению версиями осуществляется специальным сервером. В отличие от классических, в распределённых системах контроля версий центральный репозиторий не является обязательным.

4. Опишите действия с VCS при единоличной работе с хранилищем.

Сначала создаем и подключаем удаленный репозиторий. Затем по мере изменения проекта отправлять эти изменения на сервер.

5. Опишите порядок работы с общим хранилищем VCS.

Участник проекта (пользователь) перед началом работы посредством определенных 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.

6. Каковы основные задачи, решаемые инструментальным средством git?

Первая — хранить информацию о всех изменениях в вашем коде, начиная с самой первой строчки, а вторая — обеспечение удобства командной работы над кодом.

7. Назовите и дайте краткую характеристику командам 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 с текущим 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. Приведите примеры использования при работе с локальным и удалённым репозиториями.

`git push --all (push origin master/любой branch)`

9. Что такое и зачем могут быть нужны ветви (branches)?

Ветвление («ветка», branch) — один из параллельных участков истории в одном хранилище, исходящих из одной версии (точки ветвления). [3] • Обычно есть



главная ветка (master), или ствол (trunk). • Между ветками, то есть их концами, возможно слияние. Используются для разработки новых функций.

#### 10. Как и зачем можно игнорировать некоторые файлы при commit?

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.

## 6 Выводы

В результате выполнения данной лабораторной работы я приобрел необходимые навыки работы с гит, научился созданию репозитория, gpg и ssh ключей, настроил каталог курса и авторизовался в gh.

## **Список литературы**