Отчет по

Лабораторной работе №1

Выполнил: Ким Илья Владиславович  
 Группа: НФИбд-02-19

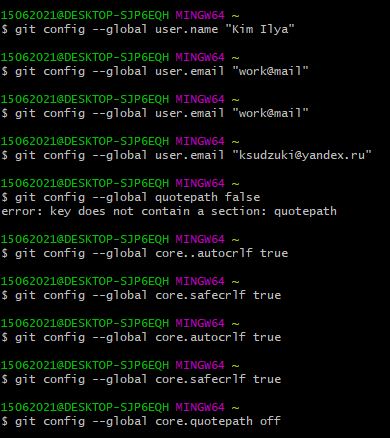
**Цель работы:**

Изучить идеологию и применение средств контроля версий.

**Этапы выполнения работы:**

1. Установка необходимых программ (Chocolatey, git Bash).

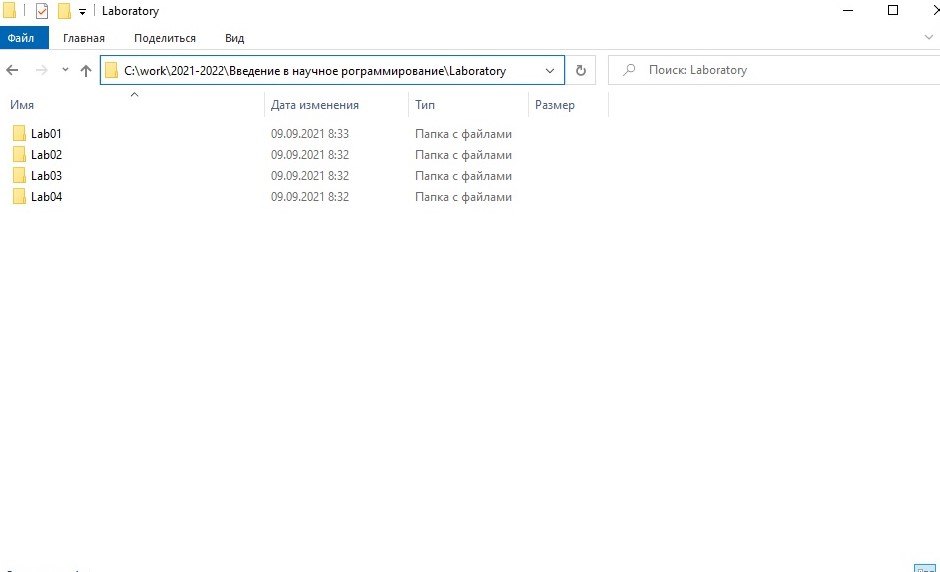
2. Настроил систему контроля версий git



3. Создал учетную запись на github и подключил к ней ssh ключ



4. Создал структуру каталога лабораторных работ



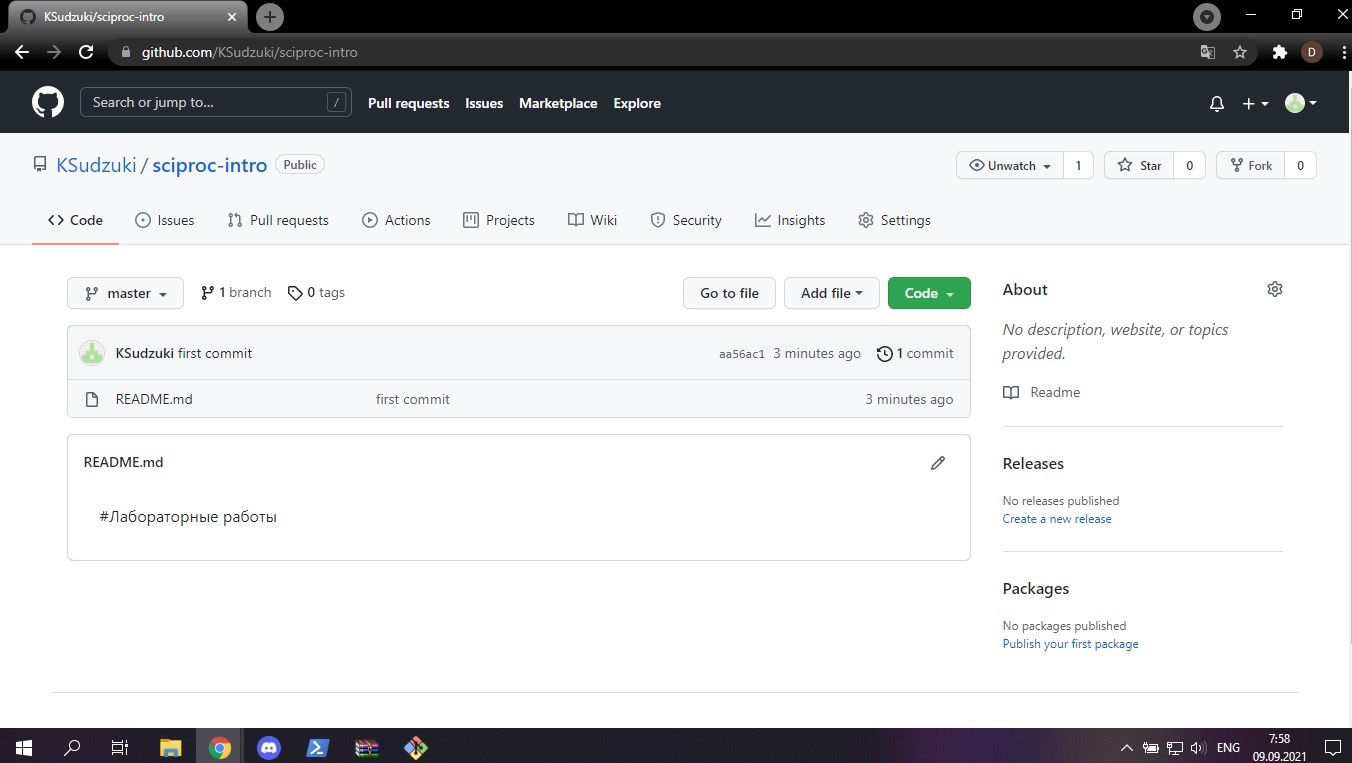
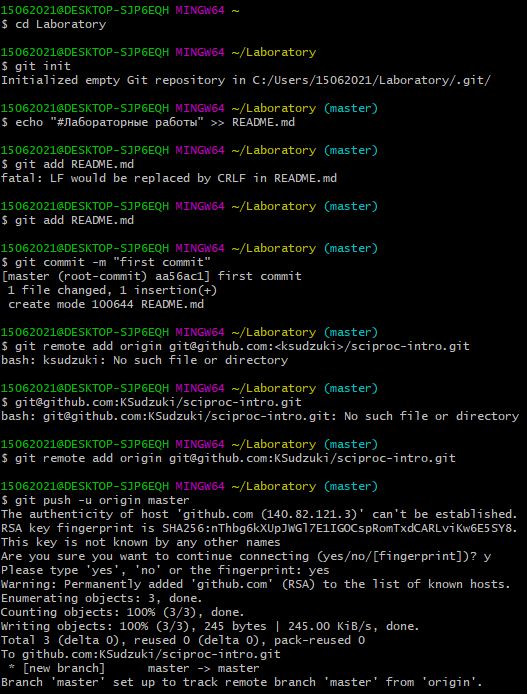
5. Создал реозиторий на github и назвал его “sciproc-intro”

* Рабочий каталог обозначил как “Laboratory”
* В Git Bash перешёл в него командой “cd laboratory”
* Инициализировал системы git командой “git init”
* Создал заготовку для файла README.md

git add README.md

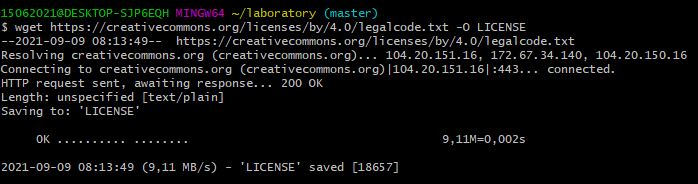
Сделал первый коммит и выложил его на github

1. git commit -m "first commit"
2. git remote add origin git@github.com:KSudzuki/sciproc-intro.git
3. git push -u origin master



6. Первичная конфигурация

* Добавил файл лицензии командой wget (Установил ее через Chocolatey)



* Посмотрел спписок шаблонов игнорируемых файлов



* Скачал шаблон



* Добавил новые файлы

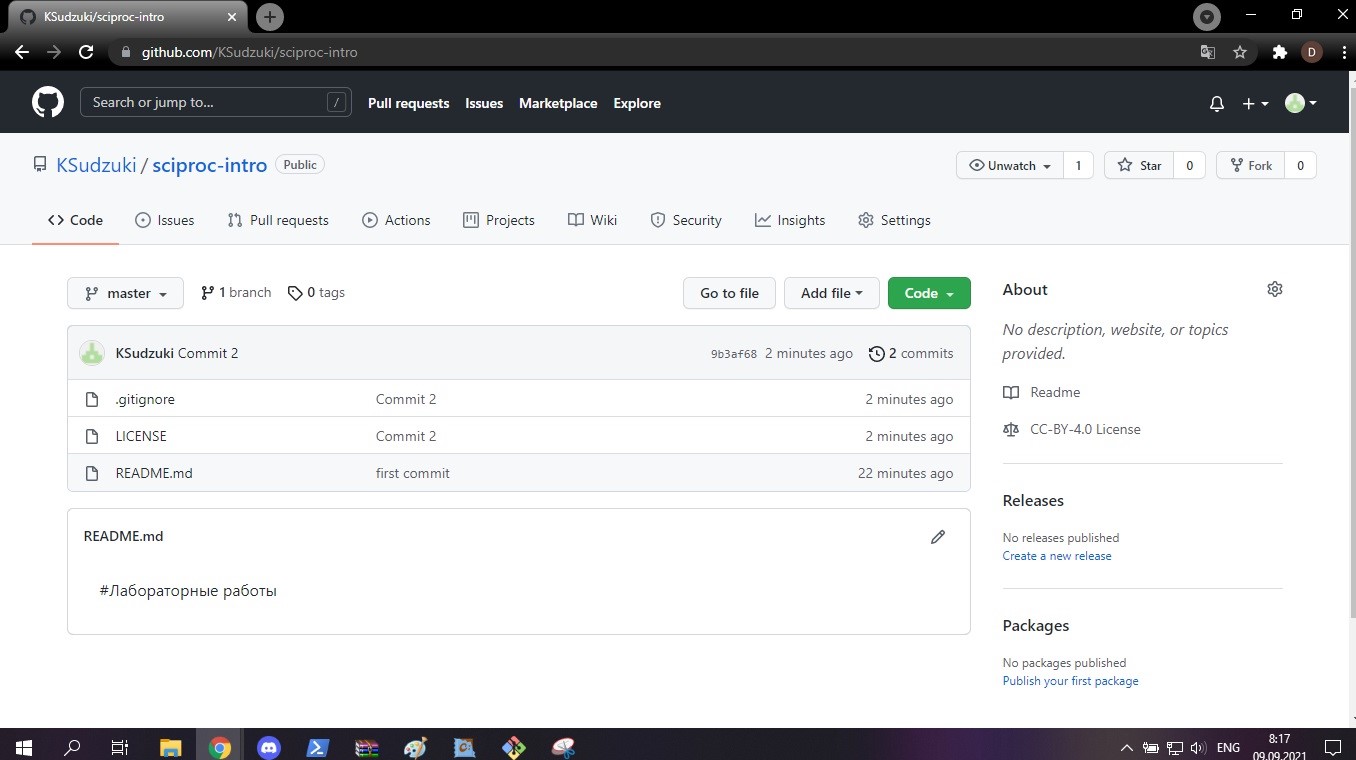
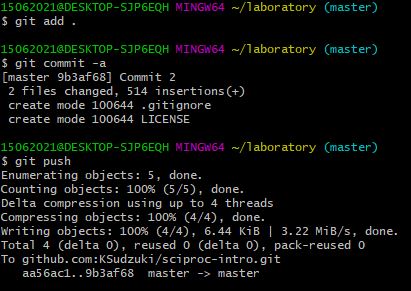
git add .

* Выполнил коммит

git commit –a

* Отправил на github

git push



7. Конфигурация git-flow

* Инициализировал git-flow

git flow init

Префикс для ярлыков установил v.

* Проверил что я на ветке develop (горит зеленым)

git branch

* Создал релиз с версией 1.0.0

git flow release start 1.0.0

* Записал версию:

Echo “1.0.0” >> VERSION

* Добавил в индекс:

git add .

git commit –am ‘chore(main): add version’

* Залил релизную ветку в основную ветку

git flow release finish 1.0.0

* Отправил данные на github

git push –all

git push –tags



Вывод:

Изучили идеологию и применение средств контроля версий git.

Контрольные вопросы:

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

Система контроля версий – это система, которая позволяет группе людей работать над одним проектом. Его основное дерево, обычно, находится на локальном или удаленном репозитории, к которому имеют доступ все участники проекта.

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

Хранилище (repository) – место, где хранятся наши файлы на удаленном или локальном VCS.  
Commit – команда, которая сохраняет сделанные изменения с внесением комментария, без отправки на репозиторий.

История – история изменений, которые были сделаны в хранилище.

Рабочая копия – место работы разработчика, до отправки в репозиторий (commit вносит изменения в рабочую копию)

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

Централизованные – это VCS с одним основным хранилищем для всего проекта. Каждый пользователь может скопировать себе необходимые ему файлы из этого репозитория, поменять, а затем добавить обратно

(Примеры: Subversion, CVS, TFS)

Децентрализованные – это VCS, где у каждого пользователя свой вариант(возможно несколько) репозитория. Есть возможность добавлять и забирать изменения из любого хранилища.

(Примеры: Git, Mercurial, Bazaar)

1. **Опишите действия с VCS при единоличной работе с хранилищем.**

**???**

1. **Опишите порядок работы с общим хранилищем VCS.**

Общее хранилище VCS служит для хранения текущих(актуальных) версий того, или иного проекта, над которым работает команда. Изначально мы добавляем туда файлы, а далее может брать их, менять, и добавлять новые версии.

1. **Каковы основные задачи, решаемые инструментальным средством git?**

Упрощение работы над групповым и индивидуальным проектом, отслеживание изменений, возврат к предыдущим версиям.

1. **Назовите и дайте краткую характеристику командам git.**

**git init** – создание основного дерева репозитория

**git pull** – получение обновлений (изменений) текущего дерева из центрального репозитория

**git push** – отправка всех произведённых изменений локального дерева в центральный репозиторий

**git status** – просмотр списка изменённых файлов в текущей директории

**git diff** – просмотр текущих изменения

**git add .** – добавить все изменённые и/или созданные файлы и/или каталоги

**git add имена\_файлов** – добавить конкретные изменённые и/или созданные файлы и/или каталоги

**git rm имена\_файлов** – удалить файл и/или каталог из индекса репозитория (при этом файл и/или каталог остаётся в локальной директории)

**git commit -am 'Описание коммита'**– сохранить все добавленные изменения и все изменённые файлы

**git commit** – сохранить добавленные изменения с внесением комментария через встроенный редактор

**git checkout -b имя\_ветки** – создание новой ветки, базирующейся на текущей

**git checkout имя\_ветки** – переключение на некоторую ветку (при переключении на ветку, которой ещё нет в локальном репозитории, она будет создана и связана с удалённой)

**git push origin имя\_ветки** – отправка изменений конкретной ветки в центральный репозиторий

**git merge --no-ff имя\_ветки** – слияние ветки с текущим деревом

**git branch -d имя\_ветки** – удаление локальной уже слитой с основным деревом ветки

**git branch -D имя\_ветки** – принудительное удаление локальной ветки

**git push origin :имя\_ветки** – удаление ветки с центрального репозитория

1. **Приведите примеры использования при работе с локальным и удалённым репозиториями.**

**???**

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

Ветви нужны для того, чтобы люди, работающие над проектом могли вести совместную работу, не мешая друг другу. В ветвях можно тестировать код, а потом, после всех проверок, отправлять обратно в основную(master) ветку.

1. **Как и зачем можно игнорировать некоторые файлы при commit?**

Игнорирование используется для того, чтобы исключить не нужные для определенных случаев файлы (ос, языка программирования или среды разработки)