**МИНИCTEPCTBO НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ**

**Федеральное государственное автономное**

**образовательное учреждение высшего образования**

**«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»**

**Кафедра инфокоммуникаций**

## **«Исследование основных возможностей Git и GitHub»**

**Отчет по лабораторной работе № 1.1**

**по дисциплине «Основы программной инженерии»**

|  |
| --- |
| Выполнил студент группы ПИЖ-б-о-21-1 |
| Коновалова В.Н. « 15» сентября 2022г. |
| Подпись студента\_\_\_\_\_\_\_\_\_\_\_ |
| Работа защищена « »\_\_\_\_\_\_\_\_\_\_\_\_20\_\_г. |
| Проверил Воронкин Р.А. \_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись) |

Ставрополь 2022

Цель работы: исследовать базовые возможности системы контроля версий Git и веб-сервиса для хостинга IT-проектов GitHub.

<https://github.com/VictoriaKonovalova/Lr1>

Ответы на контрольные вопросы

1. Что такое СКВ и каково ее назначение?

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.

1. В чем недостатки локальных и централизованных СКВ?

Главный минус централизованных СКВ – уязвимость централизованного сервера. Временное выключение сервера (пусть даже на час) останавливает работу программистов, они не могут ни сохранять новые варианты версий, ни взаимодействовать между собой. В случае же повреждения диска, на котором хранится центральная база данных, все наработки по проекту теряются безвозвратно.

С локальными системами контроля версий та же история: если все данные по проекту «лежат» в одном месте, вы можете лишиться их сразу в один момент. Также ее проблемой является основное свойство — локальность. Она совершенно не преднезначена для коллективного использования.

1. К какой СКВ относится Git?

Распределенная СКВ. Суть их работы состоит в выгрузке клиентам не только версий файлов с последними изменениями, а всего репозитория. В большинстве систем доступно для хранения данных сразу нескольких удаленных репозиториев.

1. В чем концептуальное отличие Git от других СКВ?

Главное отличие Git'а от любых других СКВ (например, Subversion и ей подобных) — это то, как Git смотрит на свои данные. В принципе, большинство других систем хранит информацию как список изменений (патчей) для файлов. Эти системы (CVS, Subversion, Perforce, Bazaar и другие) относятся к хранимым данным как к набору файлов и изменений, сделанных для каждого из этих файлов во времени. Вместо этого Git считает хранимые данные набором слепков небольшой файловой системы.

1. Как обеспечивается целостность хранимых данных в Git?

Перед сохранением любого файла Git вычисляет контрольную сумму, и она становится индексом этого файла. Поэтому невозможно изменить содержимое файла или каталога так, чтобы Git не узнал об этом. Эта функциональность встроена в сам фундамент Git и является важной составляющей его философии. Механизм, используемый Git для вычисления контрольных сумм, называется SHA-1 хеш. Это строка из 40 шестнадцатеричных знаков (0-9 и a-f), которая вычисляется на основе содержимого файла или структуры каталога, хранимого Git. Git сохраняет всё не по именам файлов, а по хешам их содержимого.

1. В каких состояниях могут находится файлы в Git? Как связаны эти состояния?

Git имеет три основных состояния, в которых могут находиться ваши файлы: изменённые, индексированные и зафиксированные.

Изменённый означает, что вы изменили файл, но ещё не зафиксировали его в своем локальном репозитории.

Индексированный - это изменённый файл, текущую версию которого вы отметили для включения в следующий коммит (для фиксации в своём локальном репозитории).

Зафиксированный означает, что файл уже сохранён в вашем локальном репозитории.

1. Что такое профиль пользователя в GitHub?

Профиль пользователя – это аккаунт, в котором человек может хранить различные проекты, версии этих проектов, предоставлять другим пользователям возможность предлагать изменения, а также самому предлагать улучшения чужих проектов, предоставлять удаленный доступ к файлам.

1. Какие бывают репозитории в GitHub?

Локальный репозиторий –это репозиторий, который хранится на нашей машине, в рабочей папке проекта. Это та самая скрытая папка .git

Удаленный репозиторий- это репозиторий, который хранится в облаке, на сторонних сервисах, специально созданных под работу с проектами git.

1. Укажите основные этапы модели работы с GitHub.

В начале создается репозиторий на GitHub. Далее создается запрос на извлечение файлов и в следствии локальная копия всего репозитория со всеми версиями файлов, которая размещается на рабочем устройстве. Потом пользователь производит различные изменения в проекте и фиксирует их в удаленный репозиторий.

1. Как осуществляется первоначальная настройка Git после установки?

Нужно ввести свое имя пользователя и адрес электронной почты, можно также проверить версию установленного продукта – это и будет первоначальной настройкой. Сделать это можно с помощью командной строки следующим образом:

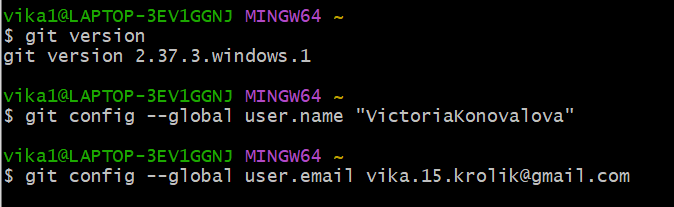


Рисунок 1.1– первоначальная настройка Git

1. Опишите этапы создания репозитория в GitHub.

Сначала надо зайти в ваш профиль. Затем создать репозиторий, для этого нужно перейти во вкладку Repositories и нажать по кнопке New.

Задать имя репозитория. Следующим шагом будет добавление описания (при необходимости).После этого необходимо выбрать режим видимости репозитория (общедоступный или приватный). Далее создаем репозиторий.

Пока проект пустой, но мы можем поместить в него наши файлы с локальной машины.

1. Какие типы лицензий поддерживаются GitHub при создании репозитория?

Публичные репозитории на GitHub часто используются для совместного использования программного обеспечения с открытым исходным кодом. Чтобы другие могли свободно использовать, изменять и распространять программное обеспечение, вам нужно будет лицензировать его.

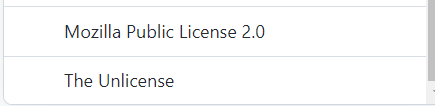
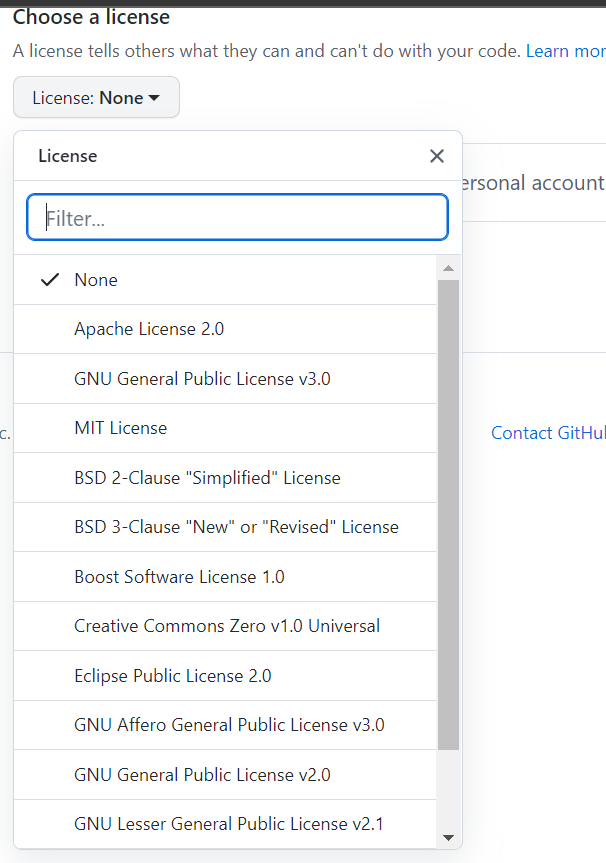
 

Рисунок 1.2– Лицензии GitHub

1. Как осуществляется клонирование репозитория GitHub? Зачем нужно клонировать репозиторий?

Клонирование репозитория локально хранит последние изменения проекта, позволяя вам разветвляться и вносить свои собственные изменения, не затрагивая сразу чужую работу. Для этого вам нужно будет загрузить Git или другое программное обеспечение, поддерживаемое Git, найти репозиторий, который вы хотите клонировать, и указать местоположение для сохранения клонированного репозитория.

1. Как проверить состояние локального репозитория Git?

Проверяем статус локального репозитория - “git status”, в ответ получаем “On branch master. nothing to commit, working directory clean”.

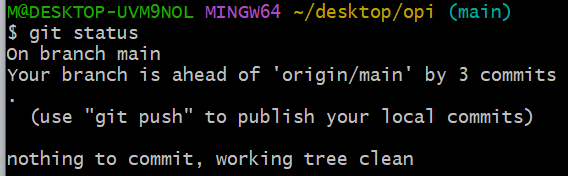


Рисунок 1.3– проверка состояния локального репозитория

1. Как изменяется состояние локального репозитория Git после выполнения следующих операций: добавления/изменения файла в локальный репозиторий Git; добавления нового/ измененного файла под версионный контроль с помощью команды git add ; фиксации (коммита) изменений с помощью команды git commit и отправки изменений на сервер с помощью команды git push ?

Важно, что коммит добавляет изменения только в ваш локальный репозиторий. Если вы хотите распространить их в исходный репозиторий на GitHub, вам нужно использовать push. В первый раз вам также необходимо отправить свою локальную ветку, потому что она не существует в удаленном репозитории. git push --set-upstream origin edit-readme В следующий раз сделать git push.

1. У Вас имеется репозиторий на GitHub и два рабочих компьютера, с помощью которых Вы можете осуществлять работу над некоторым проектом с использованием этого репозитория. Опишите последовательность команд, с помощью которых оба локальных репозитория, связанных с репозиторием GitHub будут находиться в синхронизированном состоянии. Примечание: описание необходимо начать с команды git clone.

Чтобы связать новый проект на GitHub с папкой проекта на компьютере, надо:

создать проект на GitHub (новый репозиторий) и скопировать его URL

перейти в Терминале в общую папку для проектов

клонировать проект с GitHub — git clone URL

Чтобы локальные изменения синхронизировались с удалённым репозиторием, необходимо их сначала добавить, потом закоммитить, а потом запушить.

Git add .

Git commit –m “название коммита”

Git push

1. GitHub является не единственным сервисом, работающим с Git. Какие сервисы еще Вам известны? Приведите сравнительный анализ одного из таких сервисов с GitHub.

GitKraken обладает прекрасным интерфейсом. Он ориентирован на скорость и простоту использования Git. Цель платформы — экономить время на сборку и тестирование кода. Он поддерживает GitHub, Bitbucket и Gitlab, и конечно же позволяет работать с корпоративными репозиториями с исходным кодом.

Возможности:

* Визуальное взаимодействие и подсказки;
* Поддержка нескольких профилей;
* Поддерживает кнопки отмены и повтора функции;
* Быстрый и интуитивно понятный интерфейс поиска;
* Легко адаптируется к рабочей области пользователя, а также поддерживает подмодули и Git-flow;
* Интегрируется с аккаунтами на GitHub или BitBucket;
* Горячие клавиши и многое другое.

1. Интерфейс командной строки является не единственным и далеко не самым удобным способом работы с Git. Какие Вам известны программные средства с графическим интерфейсом пользователя для работы с Git? Приведите как реализуются описанные в лабораторной работе операции Git с помощью одного из таких программных средств.

GitHub Desktop это совершенно бесплатное приложение с открытым исходным кодом, разработанное GitHub. С его помощью можно взаимодействовать с GitHub (что и не удивительно), а также с другими платформами (включая Bitbucket и GitLab).

Он дублирует функциональность сайта github.com, но при этом работает локально на компьютере разработчика, можно сразу привязать свой аккаунт.

Графический интерфейс позволяет отслеживать историю всех изменений:

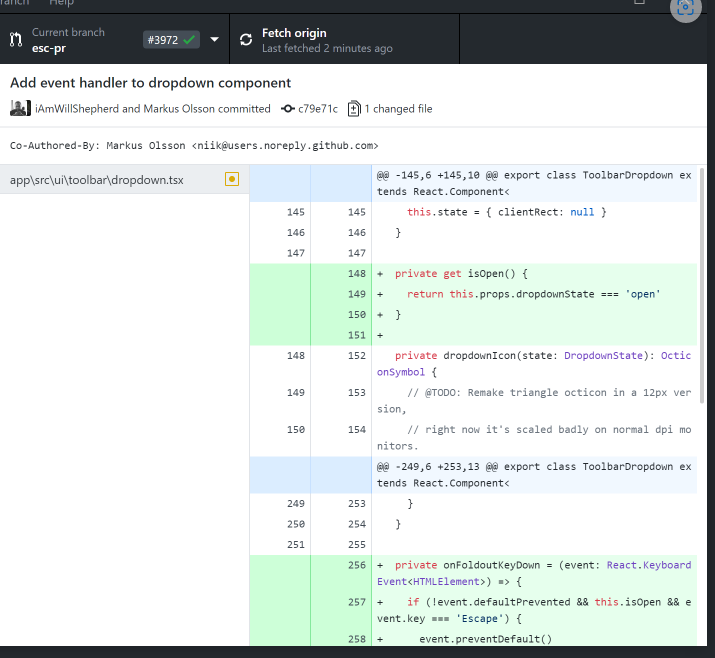


Рисунок 1.4 – История изменений в GitHub Desktop

Можно клонировать/ создать / добавить репозиторий.

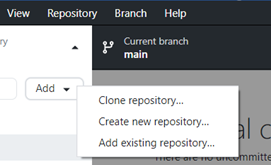


Рисунок 1.5 – Работа с репозиториями в GitHub Desktop

При нажатии на клонирование можно использовать ссылку (URL или выбрать из имеющихся на сайте репозиториев.

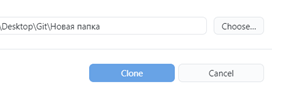


Рисунок 1.6 – Клонирование репозитория в GitHub Desktop

Создание коммита производится с помощью нажатия на кнопку «Commit to main», так же можно добавить комментарий.

Сразу после этого нажимаем на кнопку « Push origin» и отправляем на сервер.

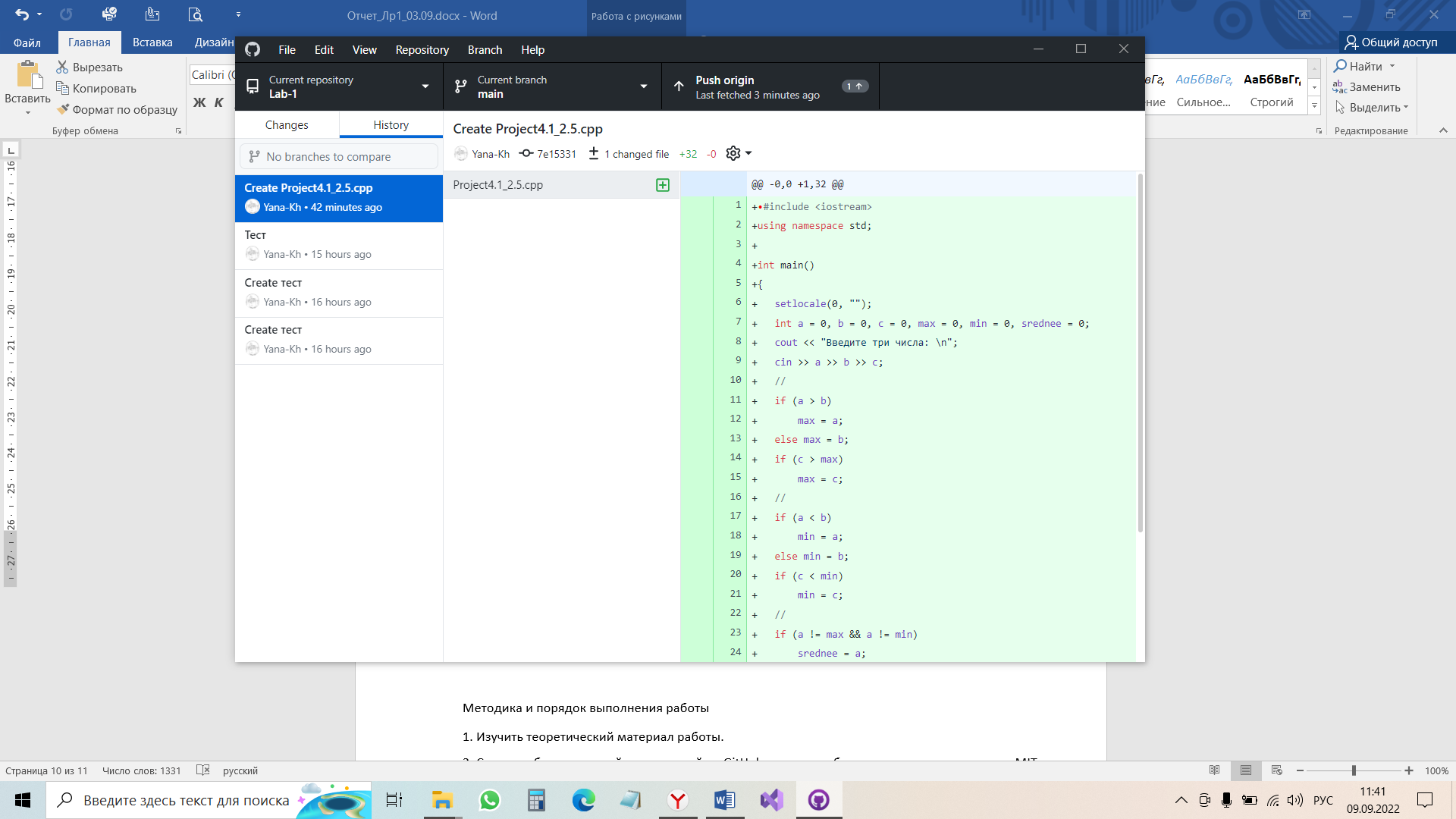
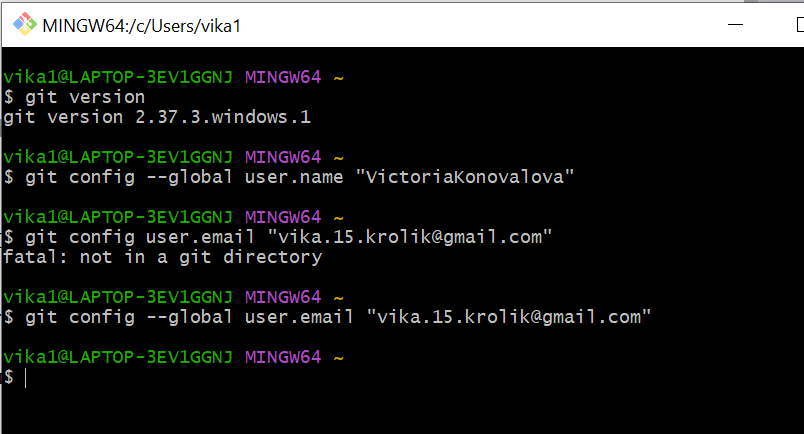


Рисунок 1.7 – Отправка изменений на сервер в GitHub Desktop

# Ход работы:

# 1. Установка и настройка Git

Рисунок 1.1 – Установка и настройка Git

2. Создание репозитория GitHub

(Создать общедоступный репозиторий на GitHub, в котором будет использована лицензия MIT и выбранный Вами язык программирования (выбор языка программирования будет доступен после установки флажка Add .gitignore).)

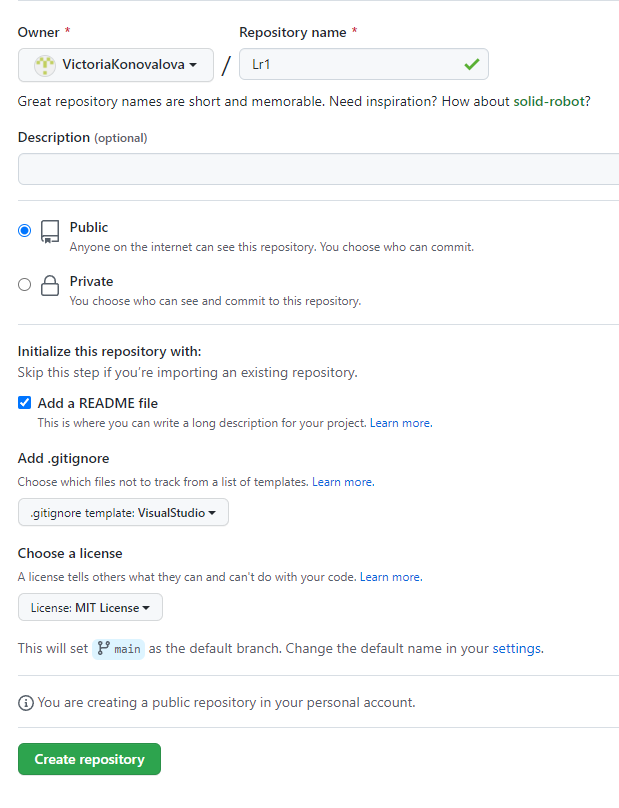


Рисунок 1.2 – Создание нового репозитория

После изучения теории, был создан общедоступный репозиторий на GitHub с использованием лицензии MIT и языка программирования С++

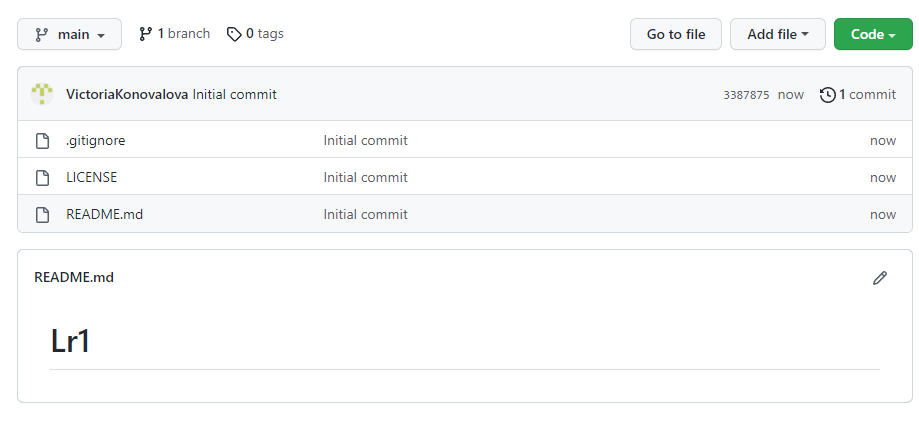


Рисунок 2.1 – Создание репозитория Lr\_1

После этого было выполнено создание локальной копии репозитория на компьютере с помощью команды «git clone».

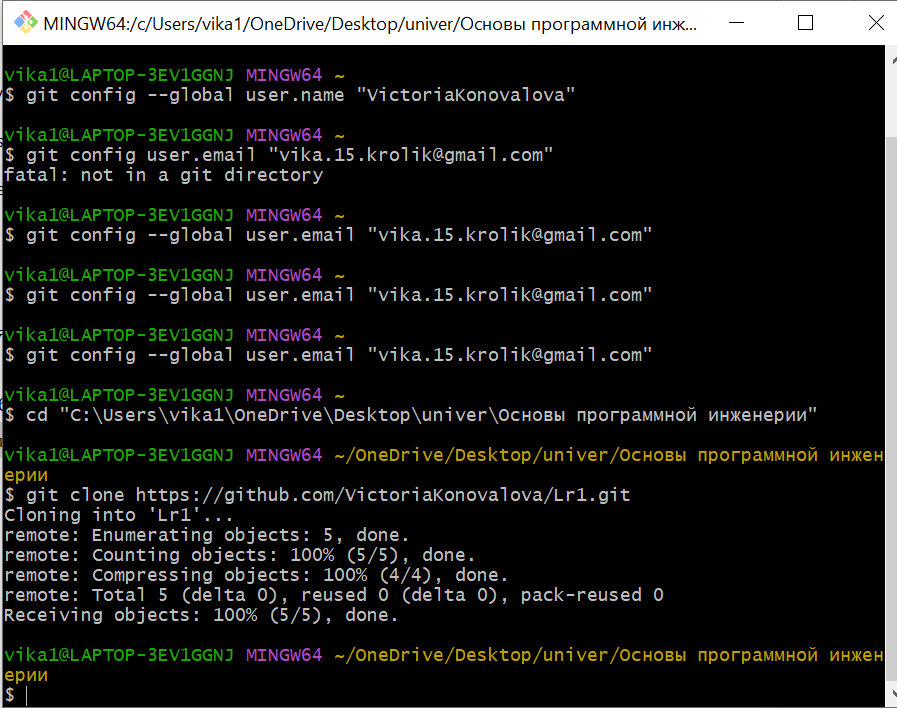


Рисунок 2.2 – Клонирование репозитория

Так как при создании репозитория был указан язык, файл .gitignore был автоматически заполнен необходимыми правилами для выбранного языка программирования и интегрированной среды разработки.

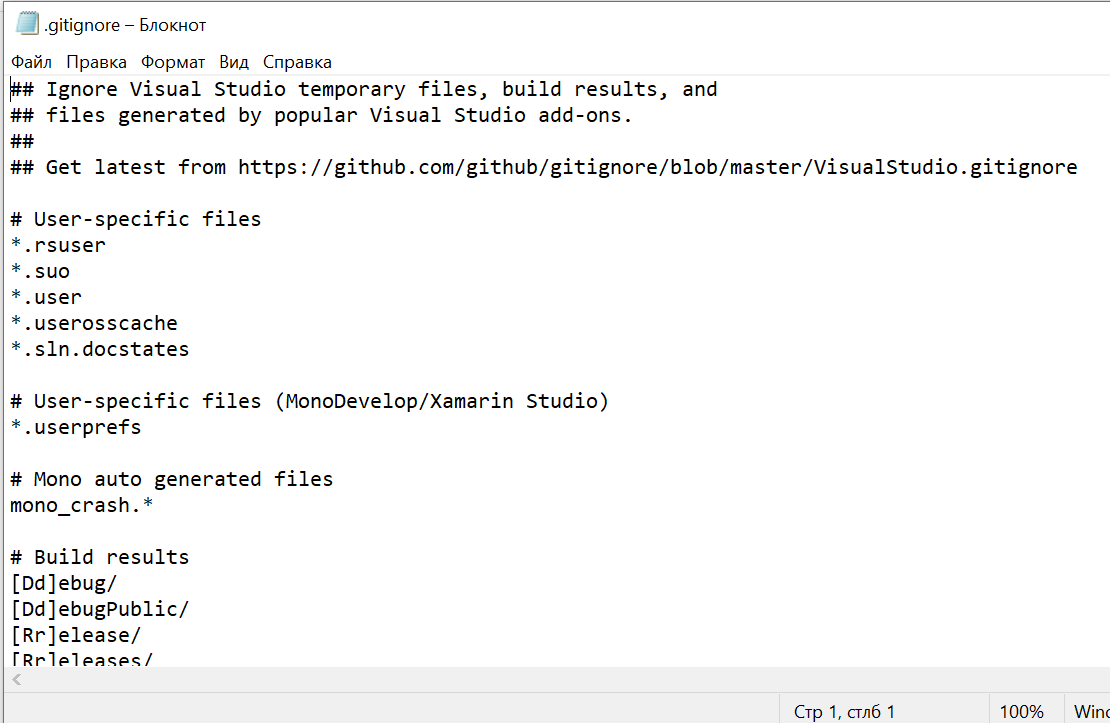


Рисунок 1.6 – Файл «.gitignore»

Далее файл README.md был дополнен информацией об имени студента и группе.

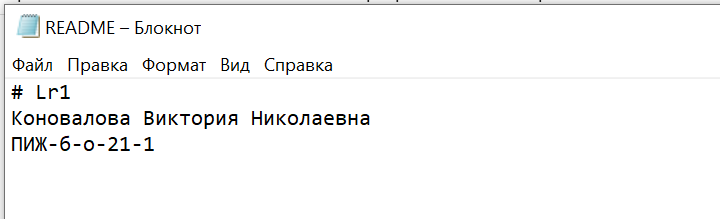


Рисунок 1.7 – Изменения в файле README.md.

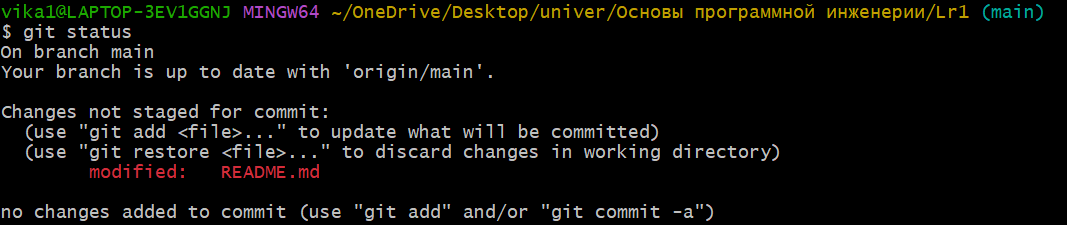


Рисунок 1.8 – Проверка статуса репозитория

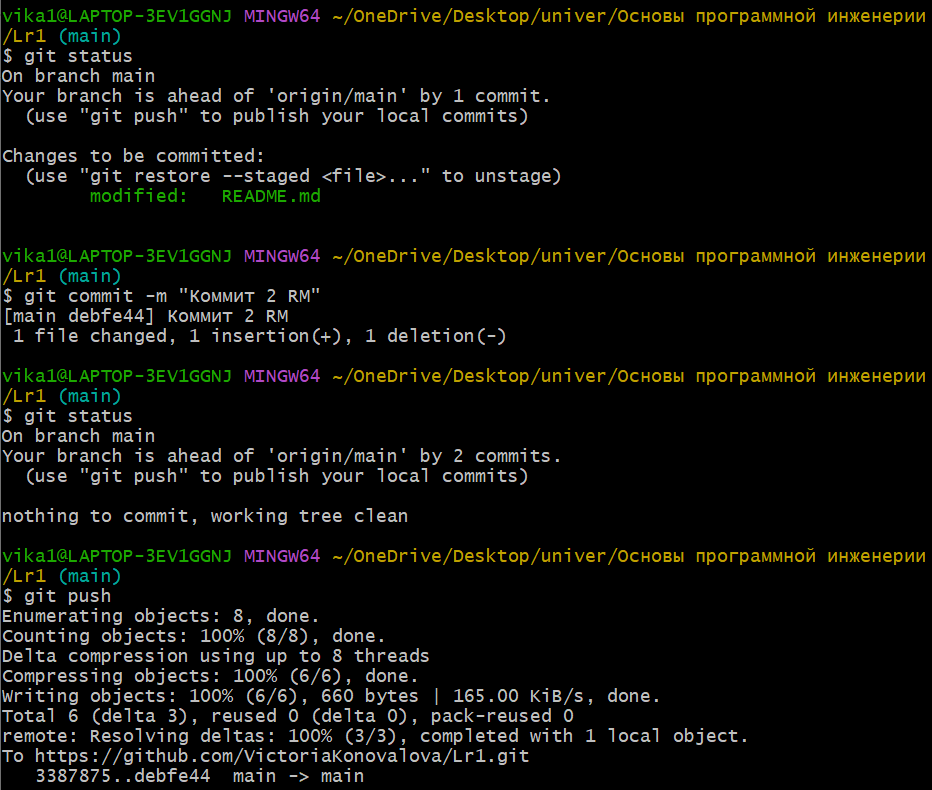


Рисунок 1.9 – Отправка изменения в удаленный репозиторий

Была написана небольшая программа на языке программирования С++. Также были зафиксированны изменения при написании программы в локальном репозитории с помощью команд «git add .», «git commit», «git push».

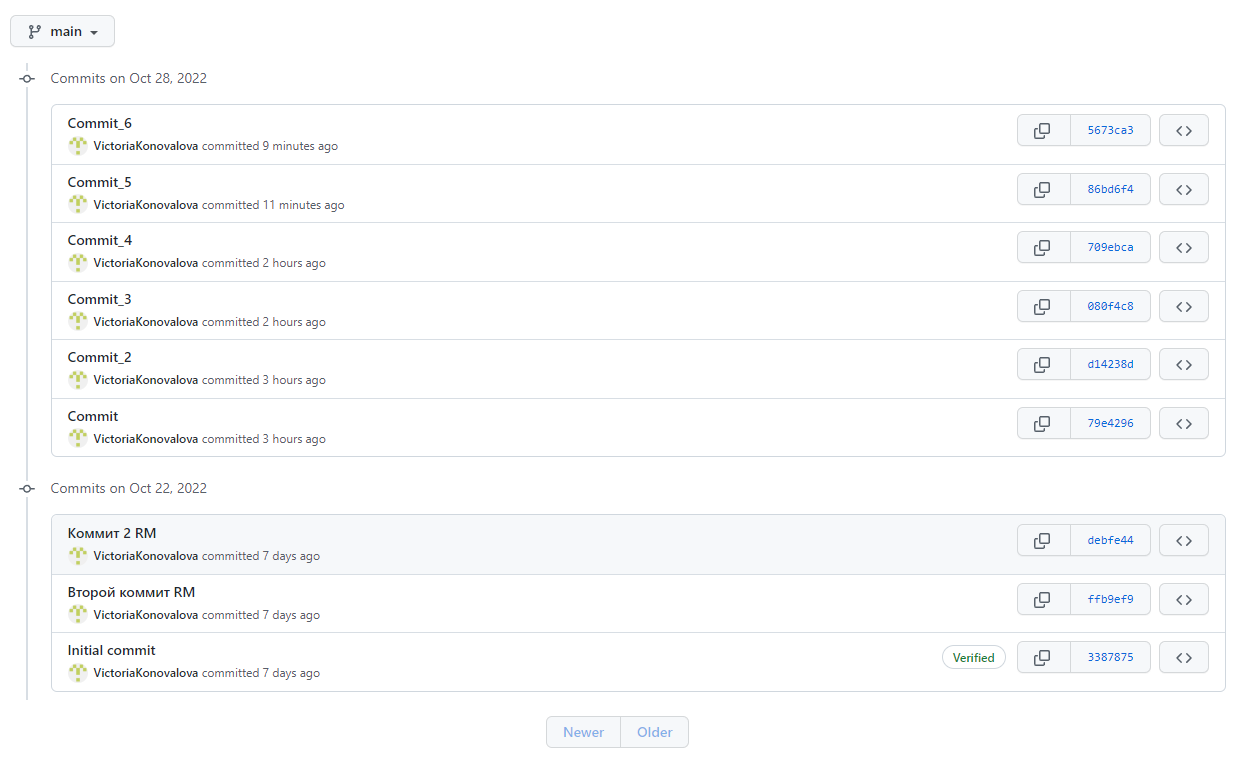


Рисунок 1.10 – Зафиксированные изменения

Далее был создан файл README\_new.txt, внесен в папку с локальным репозиторием и зафиксирован в коммите.

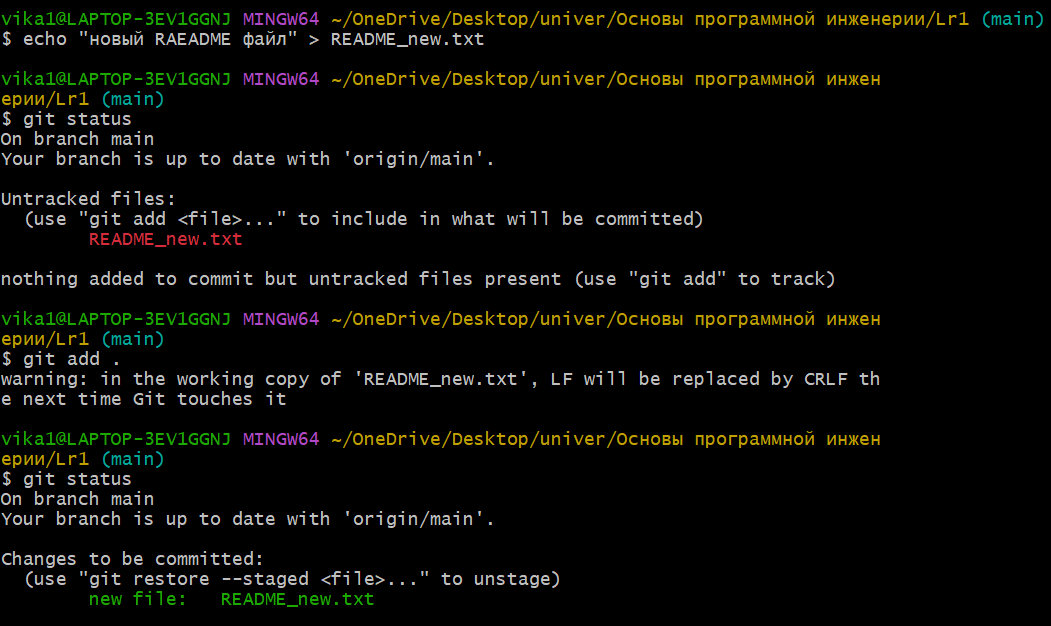


Рисунок 1.11 – Добавление файла README\_new.txt

После этого с помощью команды «git push» локальный репозиторий был отправлен на удаленный.

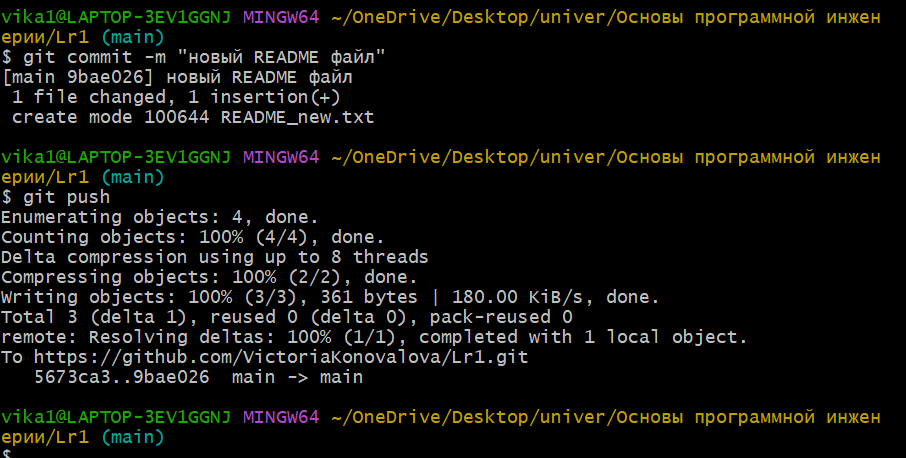


Рисунок 1.12 – распространение изменений в исходном репозитории на GitHub.

После всех проделанных действий и отправки локального репозитория в удаленный репозиторий GitHub мы можем наблюдать следующие изменения:

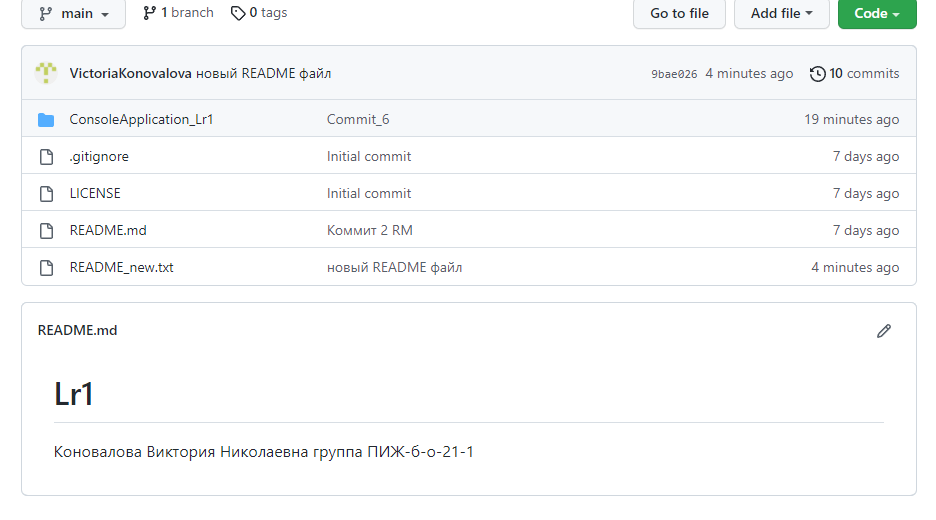


Рисунок 1.13 ­– Страница на GitHub после внесенных изменений

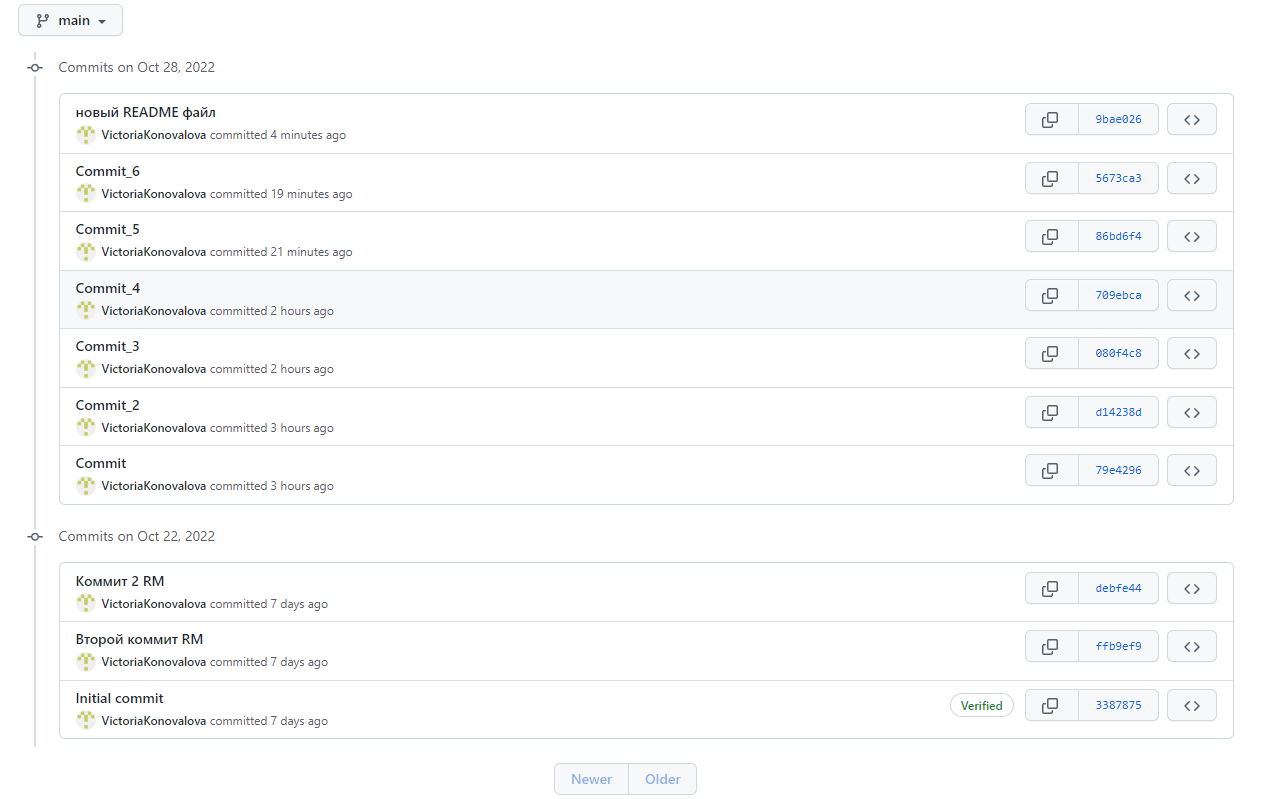


Рисунок 1.14 – Все коммиты, просмотренные в GitHub

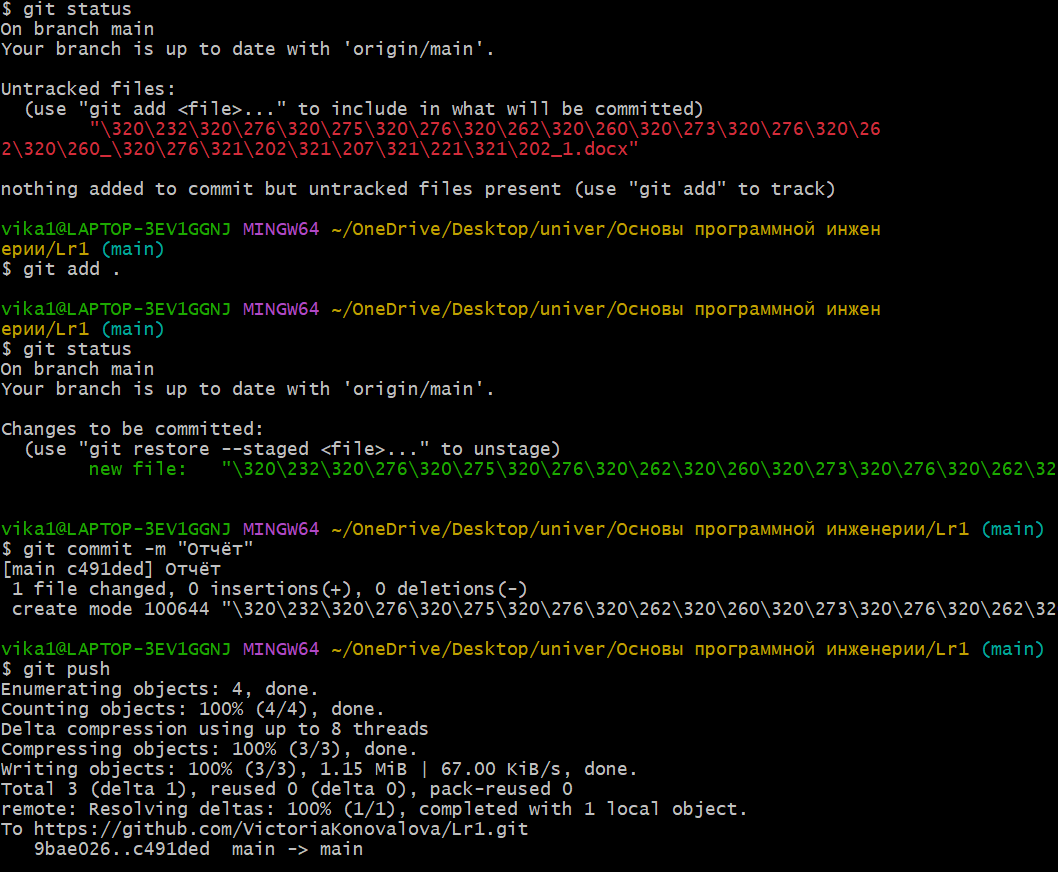


Рисунок 1.10 – Добавление начального отчета в формате .doc

**Выводы**: в ходе лабораторной работы были изучены основы работы с сервисом GitHub, а также базовые команды системы контроля версий Git.