**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 |
| Образцова М. Д. « 9 » сентября 2022г. |
| Подпись студента\_\_\_\_\_\_\_\_\_\_\_ |
| Работа защищена « »\_\_\_\_\_\_\_\_\_\_\_\_20\_\_г. |
| Проверил Воронкин Р.А. \_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись) |

Ставрополь 2022

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. Если информация потеряется при передаче или повредится на диске, Git всегда это выявит. 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. SHA-1 хеш выглядит примерно так: 24b9da6552252987aa493b52f8696cd6d3b00373 Работая с Git, вы будете постоянно встречать эти хеши, поскольку они широко используются. Фактически, в своей базе данных 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.

Базовый подход в работе с Git'ом выглядит так:

1. Вы изменяете файлы в вашем рабочем каталоге.
2. Вы выборочно добавляете в индекс только те изменения, которые вы хотите зафиксировать следующим коммитом.
3. Вы фиксируете изменения. При этом файлы в том состоянии, в котором они находятся в индексе, сохраняются в вашем репозитории навсегда в виде снимка.

Если определённая версия файла находится в репозитории, значит, она зафиксирована. Если файл был изменён и добавлен в индекс, значит, он был проиндексирован. И если файл был изменён с того момента, когда он был извлечён из репозитория, но не был добавлен в индекс, он считается изменённым.

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, зачем он нужен

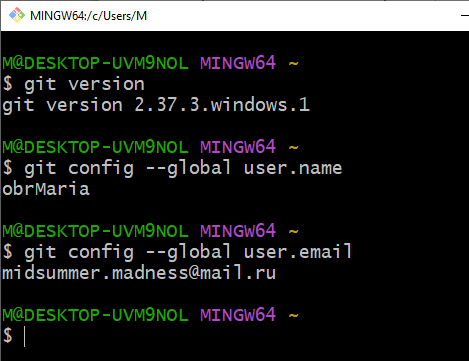
Это репозиторий, который хранится в облаке, на сторонних сервисах, специально созданных под работу с проектами git.

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:



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 ваш репозиторий действительно был с открытым исходным кодом, вам нужно будет лицензировать его, чтобы другие могли свободно использовать, изменять и распространять программное обеспечение.

Вот полный список предоставляемых лицензий:

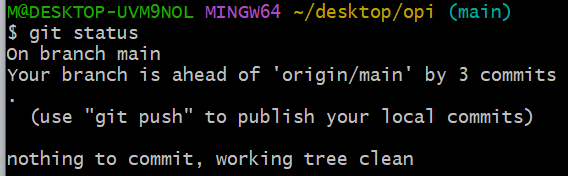
License, Academic Free License v3.0, Apache license 2.0, Artistic license 2.0, Boost Software License 1.0, BSD 2-clause "Simplified" license, BSD 3-clause "New" or "Revised" license, BSD 3-clause Clear license, Creative Commons license family, Creative Commons Zero v1.0 Universal, Creative Commons Attribution 4.0, Creative Commons Attribution Share Alike 4.0, Do What The F\*ck You Want To Public License, Educational Community License v2.0, Eclipse Public License 1.0, Eclipse Public License 2.0, European Union Public License 1.1, GNU Affero General Public License v3.0, GNU General Public License family, GNU General Public License v2.0, GNU General Public License v3.0, GNU Lesser General Public License family, GNU Lesser General Public License v2.1, GNU Lesser General Public License v3.0, ISC, LaTeX Project Public, License v1.3c, Microsoft Public License, MIT, Mozilla Public License 2.0, Open Software License 3.0, PostgreSQL License, SIL Open Font License 1.1, University of Illinois/NCSA Open Source License, The Unlicense, zLib License

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. Это можно сделать из программы командной строки или с помощью поддерживаемого программой графического интерфейса пользователя (GUI).

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”. Какие ветки создавать на новые проекты.



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 ?

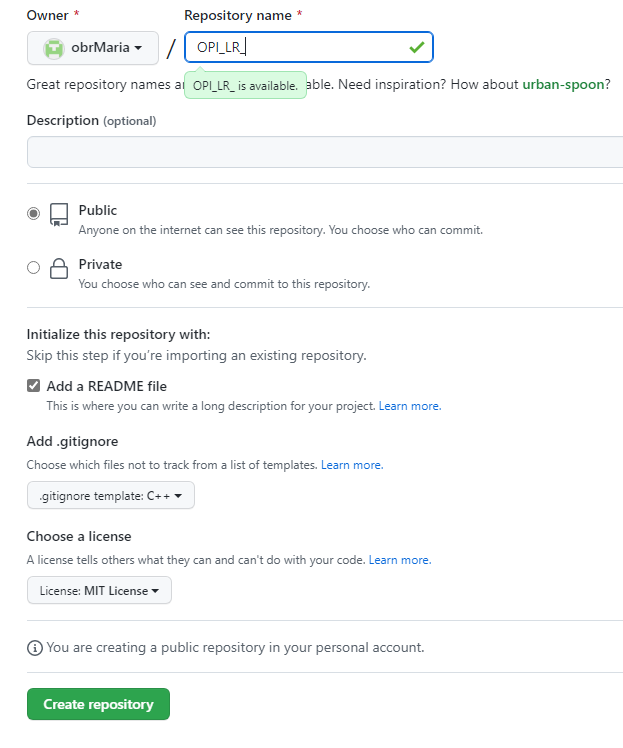
Количество коммитов обнулиться, а так как работа проводилась на локальном репозитории Вы будите иметь дело с последними версиями файлов.

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 .
2. GitHub является не единственным сервисом, работающим с Git. Какие сервисы еще Вам известны? Приведите сравнительный анализ одного из таких сервисов с GitHub.
3. 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.

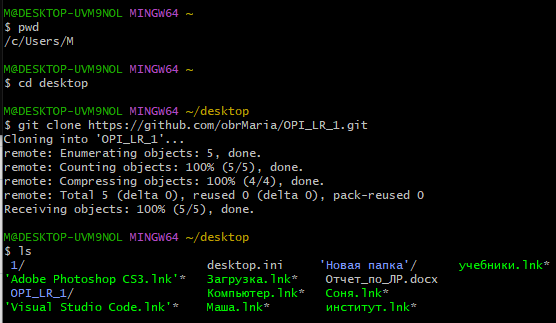
Выполнение работы

Создание репозитория на GitHub

2. 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).

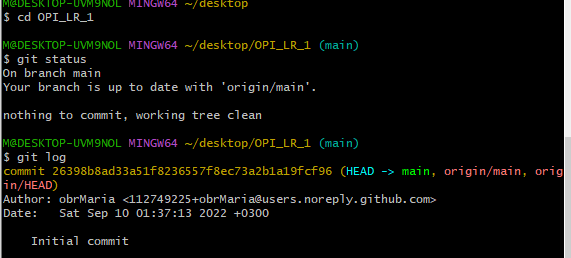


1. Выполните клонирование созданного репозитория на рабочий компьютер.

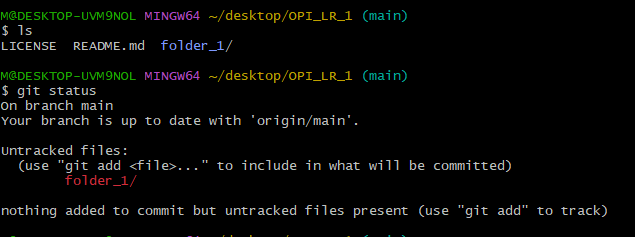


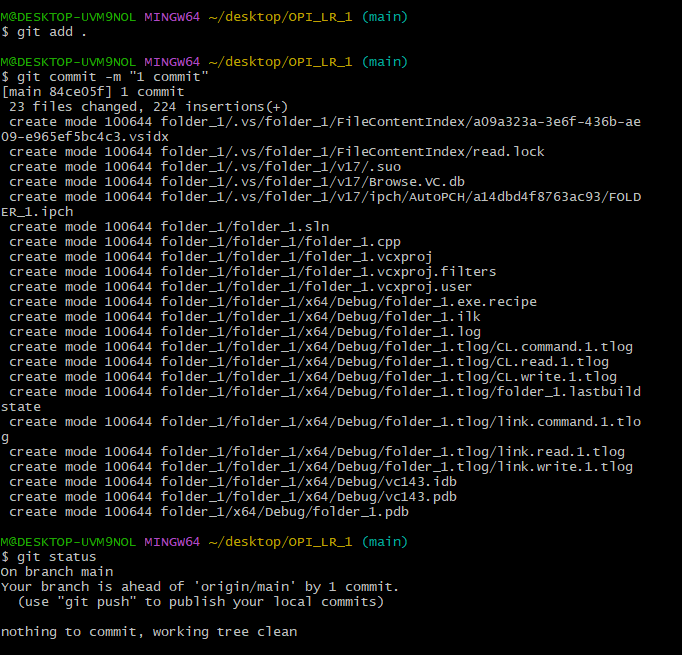
4. Дополните файл .gitignore необходимыми правилами для выбранного языка программирования и интегрированной среды разработки.

5. Добавьте в файл README.md информацию о группе и ФИО студента, выполняющего лабораторную работу.

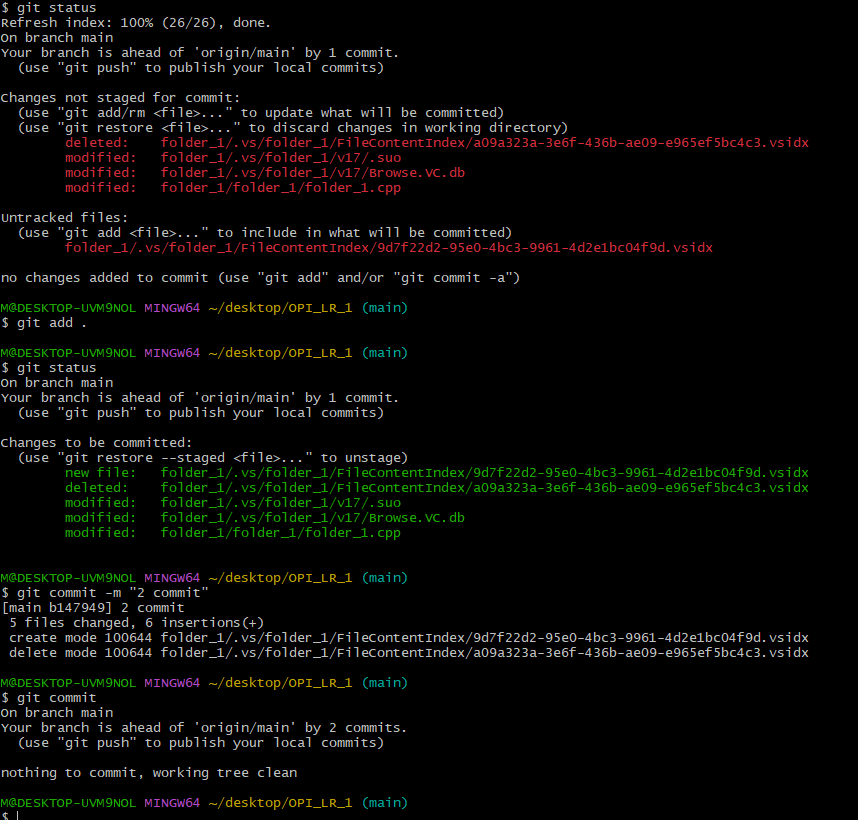


Создание коммитов.

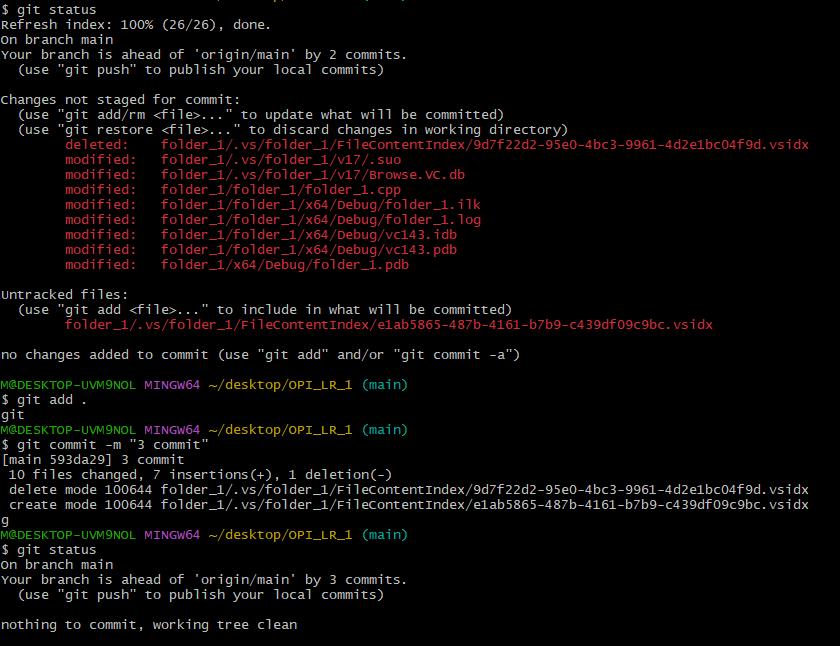




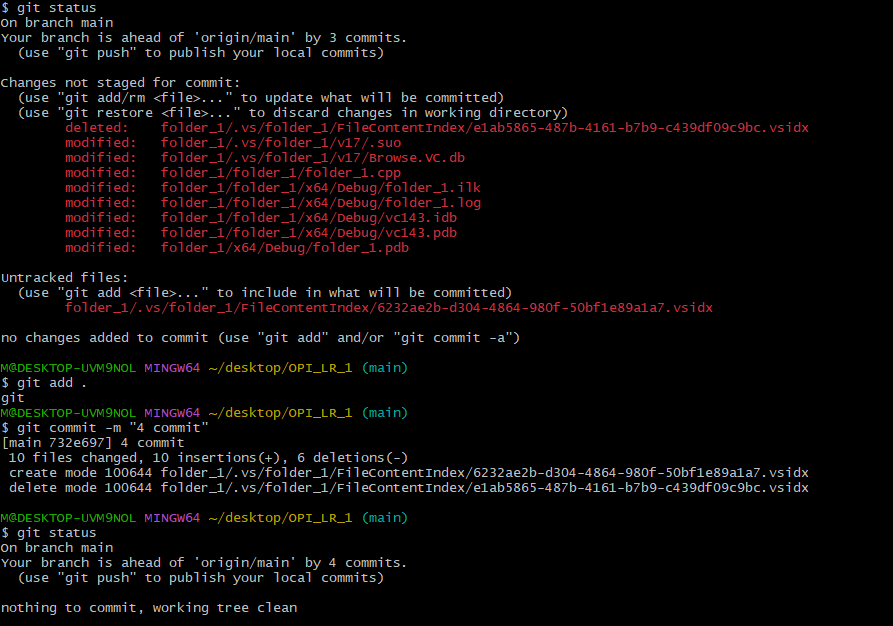
Первый коммит



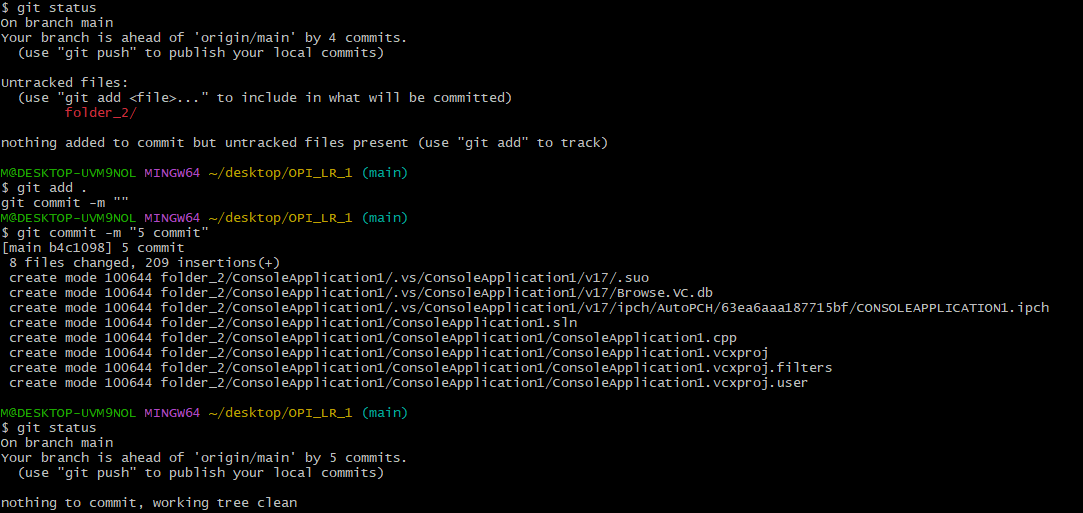
2коммит



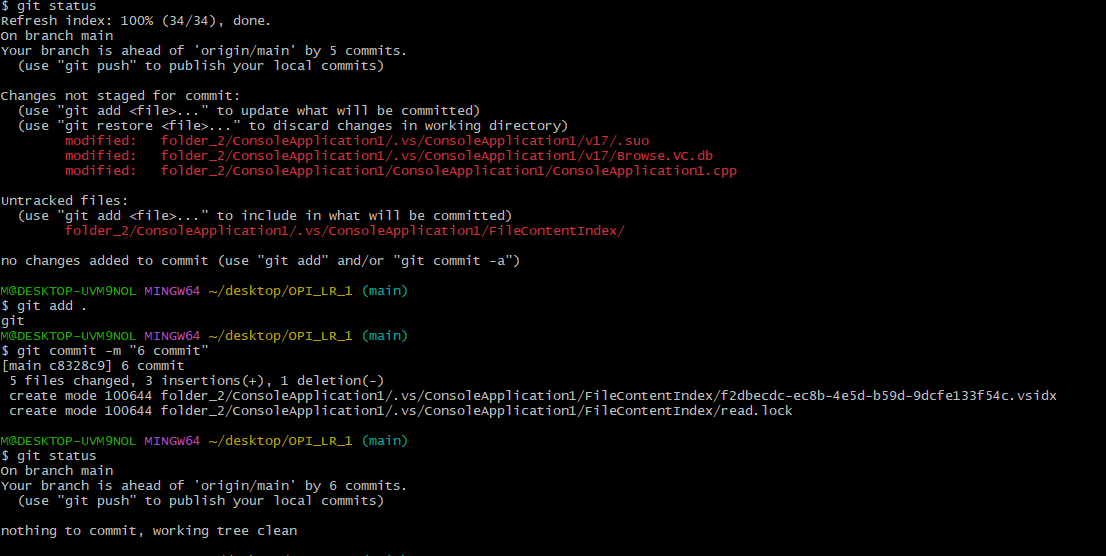
3коммит



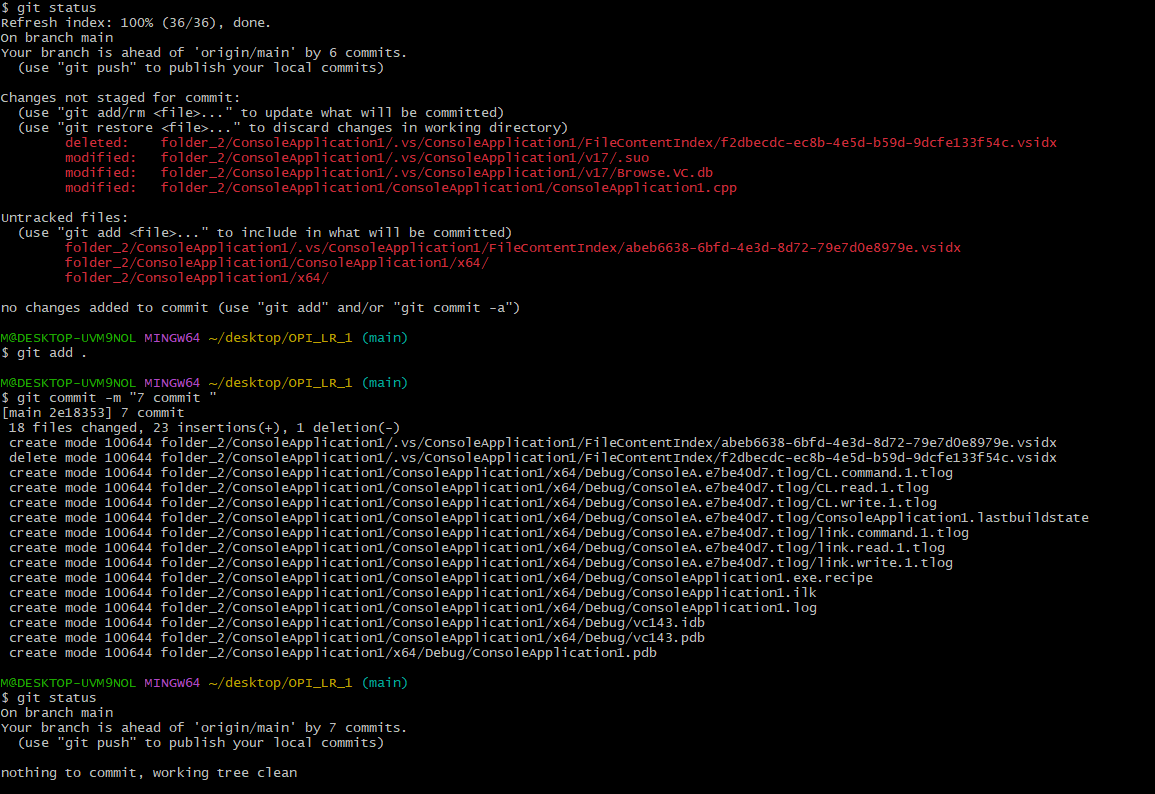
4комит



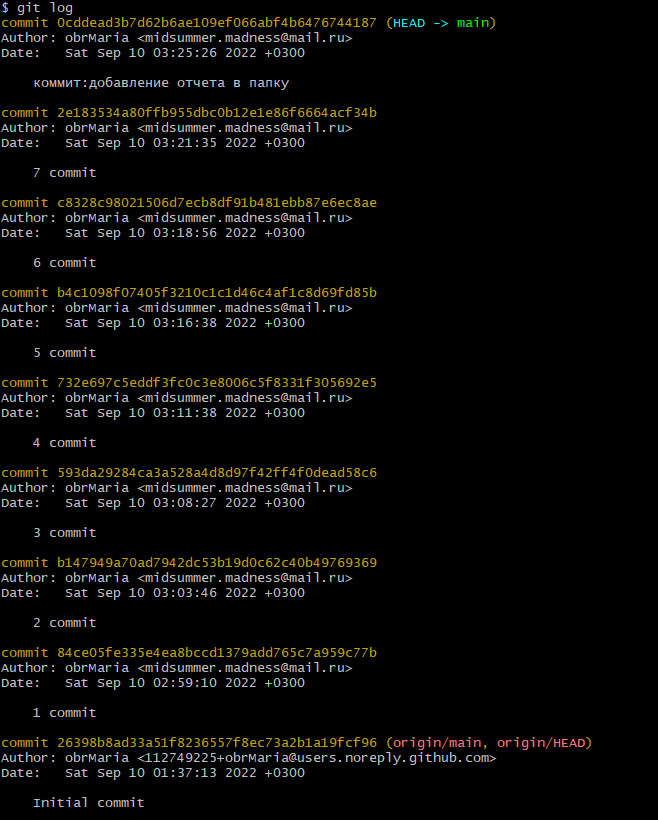
5коммит



6 коммит



7 комиит



Все коммиты

7. Добавьте файл README и зафиксируйте сделанные изменения.

8. Добавьте отчет по лабораторной работе в формате PDF в папку doc репозитория. Зафиксируйте изменения.

9. Отправьте изменения в локальном репозитории в удаленный репозиторий GitHub.

10. Проконтролируйте изменения, произошедшие в репозитории GitHub.

11. Отправьте адрес репозитория GitHub на электронный адрес преподавателя.