Федеральное государственное бюджетное образовательное учреждение

высшего профессионального образования

**Национальный исследовательский университет «МЭИ»**

**Лабораторная работа №2**

**«Система контроля версий»**

Вариант 17

Задание выполнил: **Федоров А.**

Студент группы **A-02-18**

Преподаватель: **Козлюк Д. А.**

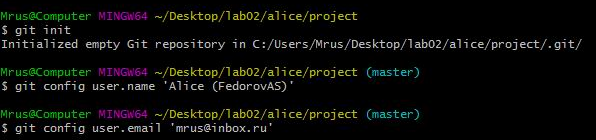
**Мохов А. С.**

Москва 2019 НИУ «МЭИ»

Ход работы.

**Инициализация репозитария и настройка Git**

Инициализируем репозитарий в текущем каталоге и настраиваем репозитарий Алисы, чтобы коммиты были от ее имени:



К приглашению командной строки добавилось (master): имя текущий ветви Git. Ветвь master используется по умолчанию.

## Создание коммитов

## Занесение файлов под контроль версий

## Собираем проект:

## 

## Вернувшись в Git Bash, просмотрим состояние рабочей копии:

## 

## На ветке master видим не добавленные файлы, которые нужно добавить, чтобы закоммитить.

## Добавим файл main.cpp в отслеживаемые (в индекс):

## 

## Еще раз просмотриваем состояние рабочей копии:

## 

## Видим, что main.cpp стал отслеживаемым

## Выполним коммит с файлом main.cpp и коротким сообщением:

## 

## Добавляем файл project.cbp в индекс и делаем коммит:

## 

## Создание коммитов с изменениями

## Заменим тело функции main() на ввод двух чисел:

## 

## Просмотрим состояние репозитария.

## 

## Видим, что main.cpp изменен.

## Делаем коммит с изменениями в main.cpp первым способом.

## 

## Добавим в тело функции вывод суммы

## 

## Делаем коммит вторым способом

## 

## Добавим вывод разности

## 

## Делаем коммит третьим способом:

## 

## Игнорирование файлов

## При запросе gitstatus заметим, что в выводе команды git status все время присутствуют каталоги bin/ и obj/.

## 

## Создаем файл .gitignore в папке репозитарияи вносим туда папки, которые нужно игнорировать

## 

## После запроса gitstatusполучаем результат, папки binи obj не отображаются

## 

## Создадим коммит с .gitignore

## 

**Просмотр истории**

**Работа с журналом репозитария**

Журнал репозитария показывает команда git log. У нее много опций, например:

git log --stat показывает файлы, измененные в коммитах;

Описывает каждый коммит

1-я строка - хэш коммита

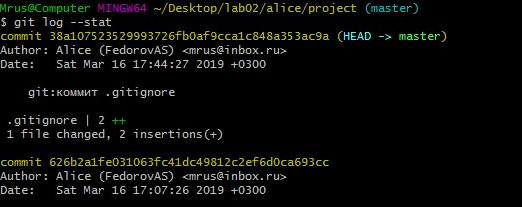
2-я строка - автор

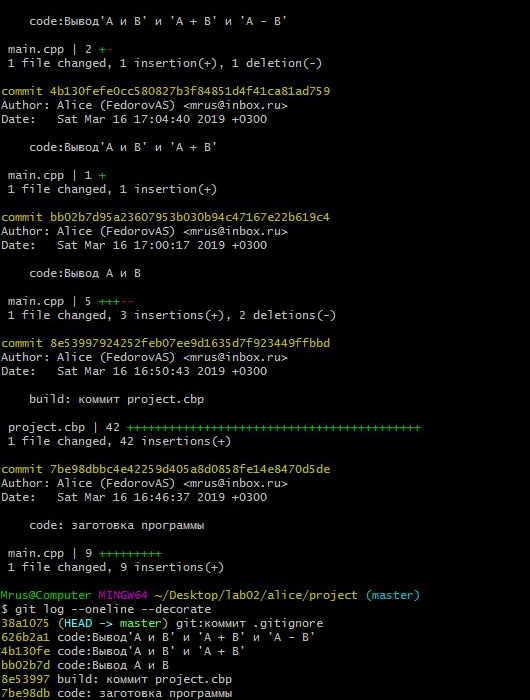
3-я строка – дата когда сделан коммит

4-я строка – комментарий к коммиту

5-я строка – имя файла и суммарное количество добавленных и удаленных строк

6-я строка – количество измененных файлов и отдельно количество добавленных и удаленных строк





## git log --oneline --decorate показывает коммиты компактно;

## 

## git log --oneline --decorate --all --graph делает то же для всех веток.

## 

## git log -- main.cpp показывает затрагивающие main.cpp;

## 

## 

## git log --grep "code:" показывает коммиты с code: в сообщении.

## 

## На основе примеров находим коммиты по теме build

## 

## Затем коммиты, затрагивающие project.cbp.

## 

## 

## Просмотр коммитов

## Просмотрим предпоследний коммит по хэшу нужного коммита

## 

## Просмотр изменений

## Просмотрим изменения между самым первым коммитом и коммитом, добавляющим вывод разности.(с помощью хэшей)

## 

## 1-4 и 15-19строки- заголовок

## 5,20-я-обозначение места изменений в файле и измененная функция

## 6,7,14-я-контекст(неизмененные строки)

## 8,9-я-удаленные строки

## 10-13 и 21,22строки- добавленные строки

## Откат изменений

## Добавим вывод произведения в main.cpp и закоммитим

## 

## 

## Предположим, необходимо отменить (откатить) этот коммит, то есть вернуться к предыдущему. Для этого воспользуемся командной git reset:

## 

## Main.cpp теперь имеет вид

## 

## Добавим над функцией main() комментарий:

## // you may type whatever you want

## 

## Уберем изменения в main.cpp другим способом — откатив этот файл к состоянию в последнем коммите (HEAD):

## 

## Main.cpp теперь имеет вид

## 

**Обмен кодом через удаленное хранилище**

**Регистрация на GitHub**

## 

**Отправка проекта на GitHub**

Создаем репозитарий под названием cs-lab02 и по инструкции настраиваем связь с хранилищем

## 

**Получение проекта с GitHub**

Предположим, к разработке проекта присоединяется Боб. Открываем новый терминал Git Bash в каталоге bob. Клонируем проект.Переходим в каталог проекта «на машине Боба» и настраиваем Git (git config) аналогично тому, как это делалось для Алисы в начале лабораторной работы.

## 

## Совместная работа над проектом без конфликтов правок

## «На машине Боба» добавляем в программу печать произведения чисел и делаем коммит.

## 

## 

## Отправляем коммит на GitHub (используя те же учетные данные, что и ранее):

## 

## «На машине Алисы» (то есть в первом терминале, в каталоге alice/project) выполним загрузку изменений:

## 

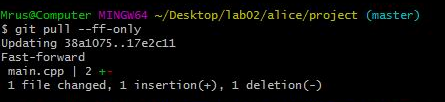
Убедимся, что в рабочей копии изменений еще не произошло.

Просмотрим историю всех веток:

## 

Как можно видеть, ветка master отстает на один коммит от ветки origin/master (версии ветки master из удаленного репозитария под названием origin, то есть на GitHub).

Продвинем ветку master к скачанной версии:



## Убедимся, что рабочая копия проекта «у Алисы» соответствует версии «у Боба».

## «На машине Алисы» добавляем в программу печать деления чисел и делаем коммит.

## 

## Отправляем коммит на GitHub (используя те же учетные данные, что и ранее):

## 

## Получаем новую версию «на машине Боба»

## 

## Разрешение конфликтов правок при совместной работе

## Предположим, Алиса решает добавить в программу печать максимума из чисел, а Боб — минимума.

## «На машине Алисы» дополним программу печатью максимума, сделаем коммит и отправьте его на GitHub.

## 

## 

## «На машине Боба» дополните программу печатью минимума, сделайте коммит и попытайтесь отправить его на GitHub.

## 

## 

## Как можно видеть, удаленный репозитарий не принимает изменений: коммит Боба основан не на последнем существующем коммите.

## «От лица Боба» загружаем коммиты из удаленного хранилища и отображаем историю всех веток.

## 

## Можно видеть, что ветка master раздвоилась. Бобу нужно переместить свой коммит поверх коммита Алисы, то есть поверх origin/master:

## 

## Однако эта команда завершается с ошибкой, сообщающей о конфликте в main.cpp.

## При введение команде git status

## 

## Git пишет, чтопроцедураrebase не завершена ипредлагает варианты действий

## 

## «На машине Боба» в CodeBlocks место конфликта будет отмечено прямо в коде.

## Приводим программу в такой вид

## 

## После того, как конфликт разрешен, добавляем файл в индекс и продолжаем прерванную операцию rebase :

## Отправляем изменения на github

## Использование веток

## Предположим, пока Боб синхронизировал изменения, Алиса решила изменить тип чисел с целых на действительные. Предполагая, что это займет время, Алиса ведет работу в отдельной ветке. На момент начала работы репозитарий Алисы не синхронизирован с GitHub, то есть последний коммит добавляет печать максимума. Все действия ведутся «на машине Алисы».

## Создадим ветку double:

## 

## Переключаемся на нее:

## 

## Заменим тип переменных a и b на double и сделаем коммит.

## 

## 

## Переключаемся на ветку master и синхронизируем ветку master «на машине Алисы» с GitHub.

## 

## Просмотрим историю всех веток

## 

## Слеваем ветку double в master:

## 

## В результате слияния образуется специальный новый коммит (merge commit), к которому Git предлагает написать сообщение в редакторе. Строки, начинающиеся с октоторпа («решетки», #), в сообщение не войдут.

## Просмотрим историю всех веток

## Путь на github https://github.com/FedorovAS/cs-lab02

## 

## Конечный результат

## 