**ПИ20-2 Лим Анастасия**

**Лабораторная работа №1 «Основы работы с СКВ в графическом режиме»**

**Цель работы**

Познакомиться на практике с основными приемами работы в современных системах контроля версий.

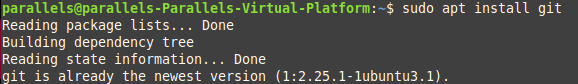
**Задания для выполнения**

1. Установить на компьютер графический клиент Git.

В качестве графического клинета Git я установила GitHub Desktop:



Но для реализации установки git я булу использовать терминал на виртуальной машине:



1. Создайте в своей домашней папке (или в любой другой на ваш выбор) каталог, который будет содержать файлы нового программного проекта.

Изображение выглядит как текст

Автоматически созданное описание

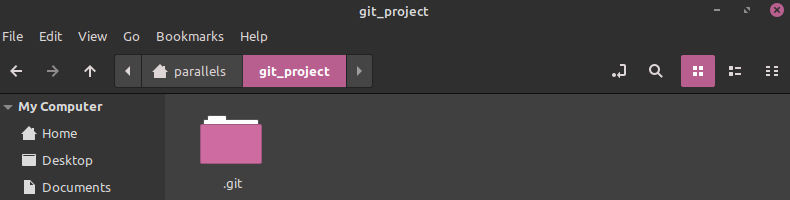
1. Выберите тематику программы, которую собираетесь написать. Язык программирования и используемые инструменты разработки сейчас не важны.

Язык Python, я выбрала задачу из егэ по информатике по работе с массивом. Необходимо найти количество чётных элементов массива, не кратных 3, заменить все нечётные элементы, кратные 3, на это количество и вывести изменённый массив.

1. Инициализируйте в этой директории репозиторий гит.



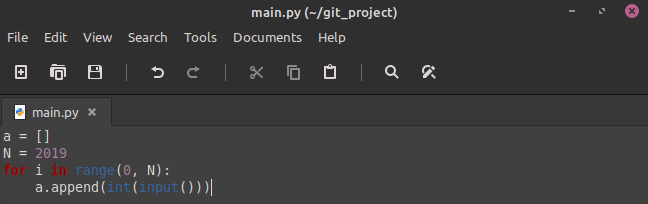
1. Обратите внимание на появление в этой папке скрытой подпапки с названием .git. Если вы ее не видите, то скорее всего, у вас отключено отображение скрытых папок.



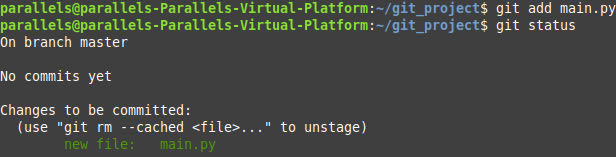
1. Создайте новый файл для исходного текста программы. Если вы используете программный фреймворк, инициализируйте его в рабочий каталог.



1. Напишите несколько строк вашей программы.

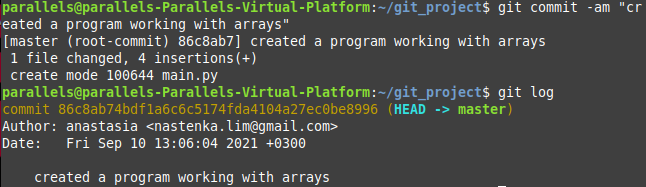


1. Добавьте файл с исходным текстом (несколько файлов, если необходимо) в индекс вашего репозитория.



1. Совершите ваш первый коммит. Напишите осмысленное сообщение коммита.

Записываем коммит с помощью команды git commit. Опция -a отвечает за выбор всех изменённых файлов, опция -m отвечает за запись сообщения коммита без потребности открытия и редактирования файла.



1. Повторите несколько раз. Каждый раз, завершая определенный этап работы, выполняйте коммит и описывайте проделанные изменения в сообщении коммита.

* Запишем строку с переменной, в которой укажем условие для нашей суммы, создадим новый коммит (created a variable and added instructions to it).
* Запишем цикл (в коммите случайно назвала функцией) с нашим основным условием отбора, сделаем коммит (created a function to run through our array).
* Запишем вывод нашего обновленного массива и коммит к обновленному состоянию (printed updated array).

Изображение выглядит как текст, устройство, счетчик, датчик

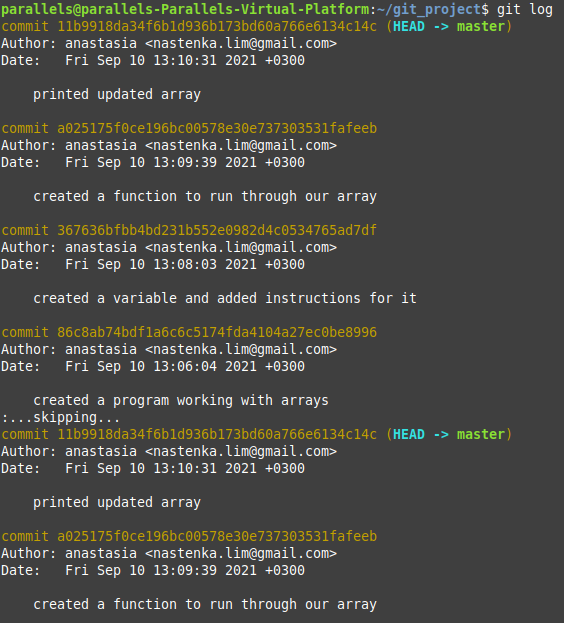
Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание

1. Просмотрите историю коммитов. Попробуйте перейти на один из прошлых коммитов. Вернитесь в актуальное состояние программы.

Просмотрим историю с помощью команды git log.



Изображение выглядит как текст

Автоматически созданное описание

С помощью команды git checkout мы перейдем к одному из предыдущих коммитов. Теперь мы находимся в состоянии detached HEAD, такое состояние означает что HEAD указывает не на вершину ветки (наш последний коммит), а на другой коммит в «центре» цепочки (в данном случае тот, на который мы переместилшись).

Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание

Возвращаемся к последнему состоянию с помощью команды git switch -.

Изображение выглядит как текст

Автоматически созданное описание

**Контрольные вопросы**

1. Опишите своими словами значение следующих терминов:
   1. рабочий каталог — каталог, в котором проинициализирован git и ведется полностью вся работа.
   2. репозиторий — коллекция всех изменений, которые были совершены на протяжении всего времени.
   3. коммит — «сохрание» текущего состояния файла в рабочем репозитории.
   4. ветка — динамический указатель на коммит, который двигается вместе с ним.
2. Ознакомьтесь с гайдом по выбранной вами программе-клиенту Git.

Я ознакомилась с гайдом по работе с Github Desktop [здесь](https://codex.so/github-start), и на [официальном сайте со всей документацией](https://docs.github.com/en/desktop/contributing-and-collaborating-using-github-desktop).

**Дополнительные задания**

1. Представьте, что вы начинаете большой раздел работы. Для изоляции изменений создайте новую ветку. Назовите ее, чтобы было понятно, что вы в ней будете делать.

С помощью команды git branch название ветки создаем новую ветку modifications, а с помощью команды git branch без опций смотрим список всех веток (сейчас мы находимся в ветке master, помечена звёздочкой):



1. Перейдите в новую ветку и сделайте несколько коммитов.

С помощью команды git checkout modifications переходим в новую ветку. Проверяем, переместились или нет, с помощью команды git branch.



Модифицируем наш код

Изображение выглядит как текст

Автоматически созданное описание

и делаем коммиты.

Изображение выглядит как текст

Автоматически созданное описание

С помощью git log просматриваем наши изменения.

Изображение выглядит как текст

Автоматически созданное описание

1. Перейдите в основную ветку и обратите внимание на состояние рабочей директории.

Так как мы перешли обратно на ветку master, все предыдущие коммиты, которые мы создали на ветке modifications, больше не показываются в логе рабочей директории.

Изображение выглядит как текст

Автоматически созданное описание

1. Создайте еще одну ветку для работы над другим направлением в вашей программе. Обычно так работают в команде, каждый участник в собственной ветке. Либо в ветках может идти параллельная работа над разными возможностями программы. В таком случае эти ветки называются тематическими.



1. Сделайте несколько коммитов во вновь созданную ветку.

Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание

1. Перейдите в основную ветку и слейте в нее первую тематическую ветку.

Переходим в основную метку с помощью команды git checkout master. Сливаем две ветки с помощью команды git merge.

Изображение выглядит как текст

Автоматически созданное описание

1. Слейте в основную ветку вторую тематическую. Если возникли конфликты слияния, разрешите их и завершите слияние.

При сливе второй ветки modifications в основную ветку мы получаем конфликт:



Так это выглядит в нашем файле:

Изображение выглядит как текст

Автоматически созданное описание

Решим конфликт слияния вручную, оставив один более корректный вариант, сделаем коммит отредактированного файла:

Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание

1. Удалите более не нужные тематические ветки. Обратите внимание в истории, что даже при удалении веток никакие коммиты не теряются.

Локально удаляем ветки neww и modifications с помощью команды git branch -d, где d — delete (удалить). Проверяем наличие веток с помощью команды git branch без каких-либо опций. Видим, что у нас осталась лишь наша основная ветка master.

Изображение выглядит как текст

Автоматически созданное описание