Семинар #1: Основы git. Практика.

Для сдачи этого и будущих ДЗ вам нужно создать репозиторий на GitLab под названием `devtools-homework`. Структура репозитория должна иметь вид:

```
├── seminar1_git/
│   ├── 1hello.sh
│   ├── 2animals.sh
│   └── ...
└── ...
```

Задача 1. Hello

Напишите `bash`-скрипт под названием `1hello.sh`, который должен делать следующее:

1. Писать на экране сообщение `Hello World`.
2. Создавать файл `hello.txt`, в котором будет записано `Hello World`.

Для того, чтобы сдать эту задачу положите скрипт в соответствующую папку в репозиторий `devtools-homework`.

Задача 2. Создайте папку с файлами

Напишите `bash`-скрипт под названием `2animals.sh`, который должен делать следующее:

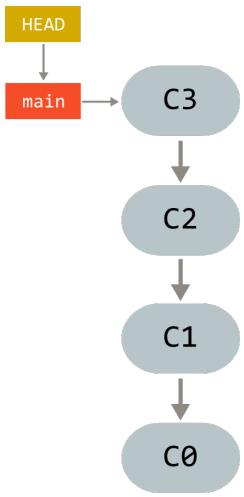
1. Создавать папку `animals` в текущей директории.
2. В этой папке создавать 2 файла: `cat.txt` и `dog.txt`.
3. В файл `cat.txt` нужно записать строку `"I am Cat"`, а в файл `dog.txt` записать строку `"I am Dog"`.

Для того, чтобы сдать эту задачу положите скрипт в соответствующую папку в репозиторий `devtools-homework`.

Задача 3. Репозиторий из четырёх коммитов

Проделайте следующие шаги:

1. Создайте новую папку `animals_repo` и перейдите в неё.
2. Инициализируйте новый пустой git-репозиторий.
3. Создайте файл `cat.txt` и добавьте коммит, содержащий этот файл.
4. Создайте файл `dog.txt` и добавьте коммит, содержащий этот файл.
5. Измените файл `cat.txt` и добавьте коммит, содержащий эти изменения.
6. Удалите файл `dog.txt` и добавьте коммит, который уже не содержит файл `dog.txt`.



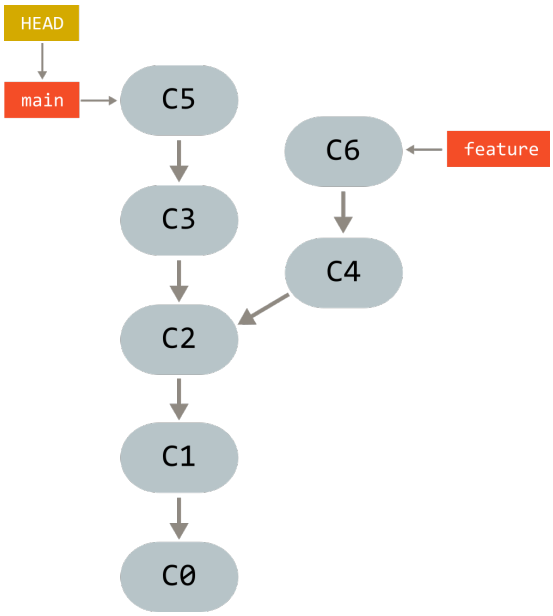
Сообщения всех коммитов должны корректно описывать происходящее. В результате должен получиться репозиторий, который будет выглядеть так, как это изображено на рисунке. Чтобы посмотреть, как выглядит граф коммитов вашего репозитория, можно использовать следующую команду:

```
$ git log --oneline --all --graph
```

Для того, чтобы сдать эту задачу, напишите `bash`-скрипт `3four_commits.sh`, который, при его выполнении, будет делать то, что описано в шагах выше и отправьте этот скрипт в репозиторий `devtools-homework`. Если вы будете запускать скрипт несколько раз, то перед каждым запуском вам нужно будет удалять сгенерированные им в прошлый раз файлы (папку `animals_repo`).

Задача 4. Две ветки

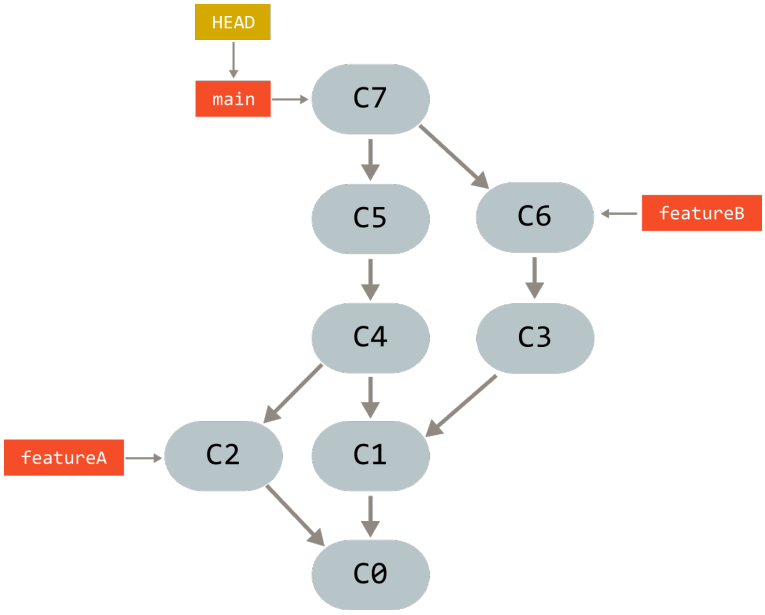
Создайте новый локальный репозиторий и добавьте в него коммиты, таким образом, чтобы граф коммитов выглядел так, как это представлено на рисунке:



В репозитории должно быть две ветки: `main` и `feature`. Порядок добавления коммитов должен соответствовать порядковым номерам, изображенным на рисунке. Сообщения коммитов должны начинаться на их обозначения, изображенным на рисунке (`C0`, `C1` и т. д.). Конкретное содержимое файлов репозитория может быть любым – на ваш выбор.
Для того, чтобы сдать эту задачу, напишите `bash`-скрипт `4two_branches.sh`, который будет создавать новый репозиторий с таким графом коммитов. После этого отправьте этот скрипт в репозиторий `devtools-homework`.

Задача 5. Граф со слияниями

Сделать всё то же самое, что и в предыдущей задаче, но граф коммитов должен выглядеть так:



Для того, чтобы сдать эту задачу, напишите `bash`-скрипт `5merges.sh`, который будет создавать новый репозиторий с таким графом коммитов. После этого отправьте этот скрипт в репозиторий `devtools-homework`.

Задача 6. Слияние с конфликтом

Для проверки числа на простоту на языке C была написана функция `is_prime`. Эта функция находится в файле `prime.c` в репозитории `mipt-hsse.gitlab.yandexcloud.net/v.biryukov/prime_calculation`. В какой-то момент файл `prime.c` выглядел следующим образом:

```c
#include <stdio.h>

int is_prime(int n)
{
    if (n < 2)
        return 0;

    for (int i = 2; i < n; ++i)
    {
        if (n % i == 0)
            return 0;
    }
    return 1;
}

int main()
{
    int a = 25263551;

    if (is_prime(a))
        printf("%i is prime\n", a);
    else
        printf("%i is NOT prime\n", a);
}
```

Над этим проектом работала команда программистов, и в определённый момент практически одновременно произошли следующие события:

- Программист Алиса заметила, что алгоритм проверки числа на простоту не оптимален и его можно ускорить, если производить итерации не до n, а до $\sqrt{n}$. Алиса создала новую ветку `alice` и добавила в него коммит, с такой оптимизацией.
- В это же самое время программист Боб заметил, что алгоритм можно ускорить, если в процессе итерирования пропускать чётные числа (однако Боб не заметил оптимизацию, которую заметила Алиса). Боб создал новую ветку `bob` и добавил в него коммит, со своей оптимизацией.
- Параллельно с этим другие участники команды продолжали развивать проект. В главной ветке `main` появился новый коммит, расширяющий набор тестов в функции `main`.

Вам нужно сделать следующее:

1. Клонировать репозиторий `prime_calculation` к себе на компьютер.
2. Объединить все изменения из веток `alice` и `bob` в ветку `main`, используя слияние – `git merge`.
3. После успешного объединения удалить локальные ветки `alice` и `bob`, если они были созданы.

Для того, чтобы сдать эту задачу, вам нужно сделать следующее:

1. Создать на GitLab новый удалённый репозиторий `prime_calculation_merge`.
2. Создать новый remote в вашем локальном репозитории, который бы указывал на новый репозиторий.
3. Загрузить коммиты из локального репозитория в удалённый.

Для проверки себя можете посмотреть граф коммитов. Граф коммитов репозитория можно посмотреть на GitLab, если на странице репозитория нажать на левой панели `Code -> Repository graph`.

Задача 7. Перебазирование с конфликтом

Та же самая задача, что и предыдущая, но вместо слияния (`git merge`) используйте перебазирования (`git rebase`). Вам нужно будет заново клонировать проект из `v.biryukov/prime_calculation` и произвести объединение веток. Но вместо слияния нужно использовать перебазирование.
Для того, чтобы сдать эту задачу, вам нужно сделать следующее:

1. Создать на GitLab новый удалённый репозиторий `prime_calculation_rebase`.
2. Создать новый remote в вашем локальном репозитории, который бы указывал на новый репозиторий.
3. Загрузить коммиты из локального репозитория в удалённый.

Задача 8. Просмотр репозитория библиотеки stb

STB – это набор header-only библиотек на C/C++, распространяемых в виде исходного кода. Библиотека предоставляет минималистичные решения для распространённых задач: загрузки и сохранения изображений в разных форматах (`.jpg`, `.png` и другие), работы со шрифтами, декодирования аудио и обработки данных. Её ключевые преимущества – простота интеграции, отсутствие внешних зависимостей и удобство использования. STB идеально подходит для быстрого прототипирования, небольших проектов и сценариев, где не требуются сложные специализированные библиотеки.

Репозиторий проекта можно найти на GitHub: `github.com/nothings/stb`. Вам нужно клонировать этот репозиторий себе: Просмотрите всю историю коммитов с помощью команды:

```
$ git log --oneline --all --graph
```

Удобнее смотреть в файле:

```
$ git log --oneline --all --graph > history.txt
```

Выполните команды, которые бы печатали на экран следующее (по одной команде на пункт):

1. Все изменения в проекте, которые были проделаны от текущего коммита (`HEAD`) к родительскому коммиту.
2. Информацию о последних пяти коммитах, а также изменения сделанные в каждом из этих пяти коммитов.
3. Информацию о всех коммитах, сделанных с начала года.
4. Информацию о всех коммитах, в сообщении которых есть подстрока `"jpeg"`.
5. Информацию о всех коммитах, которые добавляли изменения, в которых содержится подстрока `"jpeg"`.
6. Информацию о всех коммитах, в которых менялся файл `stb_textedit.h`.
7. Информацию о всех изменения (diff) файла `stb_textedit.h`.
8. Информацию о всех коммитах, в которых изменялась функция `stbi_jpeg_load` из файла `stb_image.h`.
9. Информацию о всех авторах и количество коммитов, сделанных каждым автором.
10. Информацию о том, кто последним исправил каждую строку в диапазоне от 900-й до 950-й в файле `stb_image.h`.

Для того, чтобы сдать эту задачу, напишите `bash`-скрипт `8stb.sh`, в котором будут хранится все эти команды. После этого отправьте этот скрипт в репозиторий `devtools-homework`.

Задача 9. Задание на совместную работу с репозиторием

Сделайте задание по адресу `mipt-hsse.gitlab.yandexcloud.net/dev_tools_2025/practice_1`.