Семинар: Git. Домашнее задание.

Задача 1. Простой проект

В папке `image_project` содержится исходный код простой программы, которая создаёт изображение. Можно скомпилировать эту программу командой:

```
g++ main.cpp image.cpp
```

Если запустить получившийся исполняемый файл, то программа создаст изображение `result.ppm`. Открыть изображение такого формата можно любой продвинутой программой для просмотра изображений, например, можно использовать IrfanView.

1. Создайте git-репозиторий для этого проекта, используя `git init`.
2. Сделайте первый коммит в этот репозиторий. Коммит должен содержать все файлы исходного кода проекта (`image.h`, `image.cpp` и `main.cpp`).
3. Измените файл `main.cpp` так, чтобы на изображении рисовался ещё один круг. Добавьте это изменение в репозиторий, сделав ещё один коммит.
4. Добавьте в класс `Image` метод `getNumberOfPixels`, который будет возвращать количество пикселей изображения. Сделайте соответствующие изменения в файлы `image.cpp` и `image.h` и добавьте эти изменения в репозиторий.
5. Добавьте в этот проект поддержку системы сборки Cmake. Добавьте файл `CMakeLists.txt` и соберите файл в новой папке `build`. Добавьте изменения в репозиторий. Коммит должен состоять только из одного файла (`CMakeLists.txt`). Содержимое папки `build` и прочие файл добавлять в репозиторий не надо.
6. Создайте файл `.gitignore` и добавьте туда правила для игнорируемых файлов. Система контроля версий должна игнорировать папку `build`. Добавьте файл `.gitignore` в репозиторий, сделав ещё один коммит.
7. Выполните команду `git log` чтобы увидеть информацию о ваших коммитах.
8. Класс `Image` может считывать изображения в формате `ppm`. Хочется добавить в проект изображение и изменить файл `main.cpp` так, чтобы программа считывала это изображение. Однако, также не хочется терять тот код, который сейчас находится в файле `main.cpp`.
   Поэтому создадим новую побочную ветку, чтобы вносить изменения в неё. Используйте `git branch` и создайте ветку под названием `ppm`.
9. Выполните команду `git branch` чтобы посмотреть на ваши ветки. В данный момент у вас должно быть 2 ветки `main` и `ppm` (главная ветка по умолчанию может называться `main` или `master`). Та ветка на которой вы находитесь должна быть помечена звёздочкой. Вы должны находиться на ветка `ppm`. Если это не так, то перейдите на неё с помощью `git switch`.
10. Создайте новую папку `ppm_examples` и скопируйте в него файл `files/zlatoust1910.ppm`. Измените файл `main.cpp` так, чтобы он считывал этот файл, рисовал на нём кружок и сохранял результат в файл `result.ppm`. Сохраните изменения, сделав коммит в ветку `ppm`.
11. Вернитесь обратно на ветку `main` с помощью `git switch`. Изменения, сделанные в предыдущем пункте должны исчезнуть (их можно вернуть, если снова перейти на ветку `ppm`).
12. Теперь хочется добавить поддержку изображений в формате `jpg`. Эти изменения нужно сделать в новой ветке под названием `jpg`. Создайте эту ветку и перейдите на неё.
13. Для загрузки изображений в формате `jpg` будем использовать библиотеку stb. Эта библиотека и пример её использования лежит в папке `files/stb`. Скопируйте файлы из этой папки в папку проекта. Измените класс `Image` так, чтобы он умел считывать файл в формате `jpg` (используйте библиотеку stb). Скопируйте файл `files/stb/zlatoust1910.jpg` в папку `jpg_examples` проекта. Сохраните все эти изменения в ветке `jpg`.

14. Вернитесь обратно на ветку `main` с помощью `git switch`. Изменения, сделанные в предыдущем пункте должны исчезнуть (их можно вернуть, если снова перейти на ветку `jpg`).

15. Добавьте метод `convertToGrayscale` в класс `Image`. Этот метод должен делать всё изображение чёрно-белым. Чтобы это сделать, усредните компоненты цвета каждого пикселя. Протестируйте работу этого метода в файле `main.cpp`. (Сделайте черно-белой картинку с тремя отрезками и двумя кругами). Сохраните изменения в репозиторий, сделав коммит в ветку `main`.

16. Посмотрите на граф, который образуют ваши ветки, используя:

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

17. Теперь нужно добавить метод `convertToGrayscale` в ветку `ppm`. Для этого перейдите в ветку `ppm` с помощью `git switch` и используйте `git merge` чтобы добавить изменения из ветки `main` в ветку `ppm`.

18. Также нужно добавить метод `convertToGrayscale` в ветку `jpg`. Но теперь будем использовать `rebase`. Для этого перейдите в ветку `jpg` с помощью `git switch` и используйте `git rebase` чтобы добавить изменения из ветки `main` в ветку `ppm`.

19. Посмотрите на граф, который образуют ваши ветки, используя:

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

20. Соедините теперь все 3 ветки (`main`, `ppm` и `jpg`) в одну ветку `main`. Итоговый класс `Image` должен уметь работать с файлами в форматах `ppm` и `jpg`. В файле `main.cpp` нужно считывать/изменять/сохранять как `ppm` изображение, так и `jpg` изображения.

21. Удалите ветки `ppm` и `jpg`.

22. Посмотрите на граф, который образуют ваши ветки, используя:

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

23. Чтобы сдать задание нужно загрузить локальный репозиторий этого проекта на гитхаб. Для этого нужно сделать следуюшее:

    (a) Создать новый пустой репозиторий на гитхабе

    (b) Добавить ссылку на удалённый репозиторий в ваш локальный репозиторий с помощью команды:

    ```
    git remote add origin "URL вашего нового репозитория на гитхаб"
    ```

    (c) Обновить репозиторий на гитхаб:

    ```
    git push -u origin main
    ```