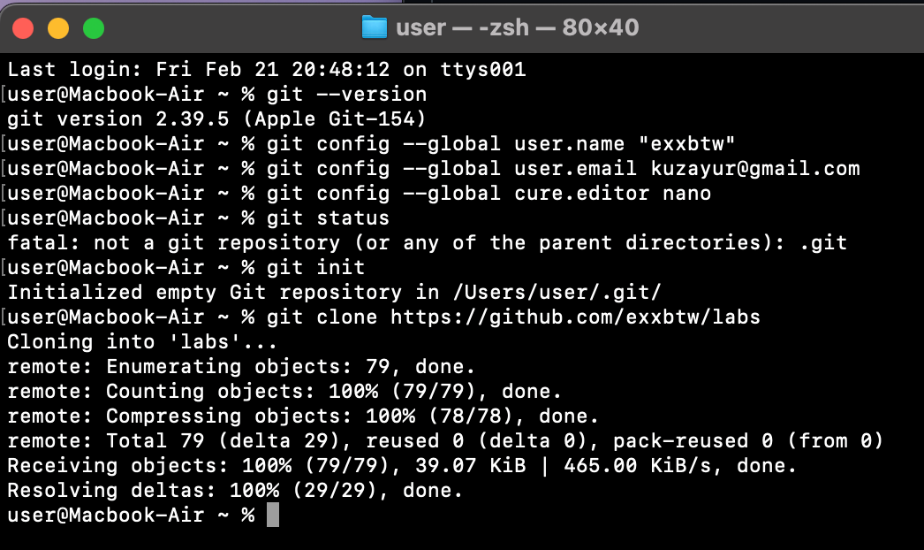
На тройку

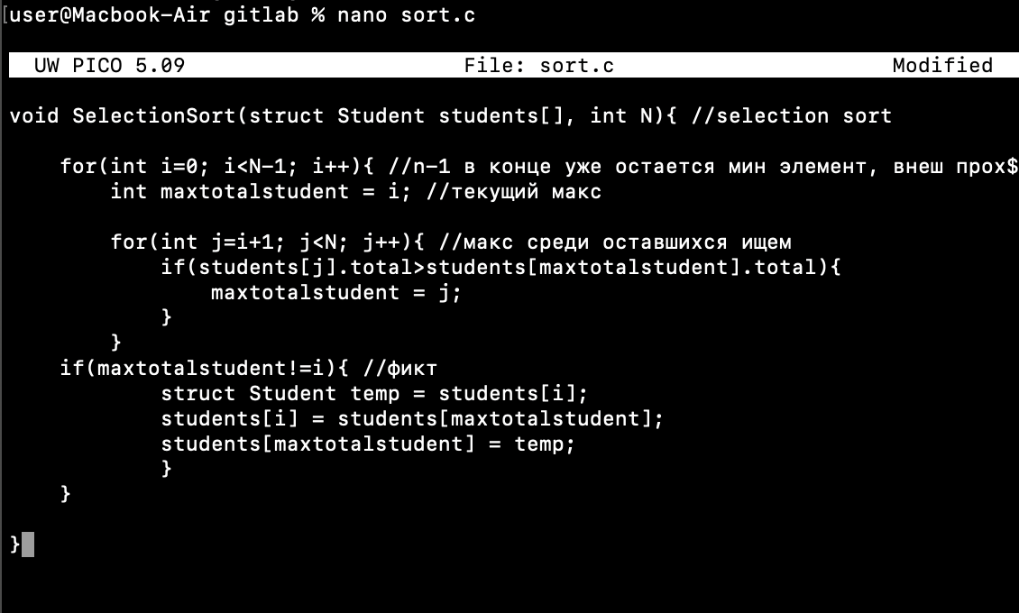
Проверяем версию Git и логинимся, клонируем репозиторий:



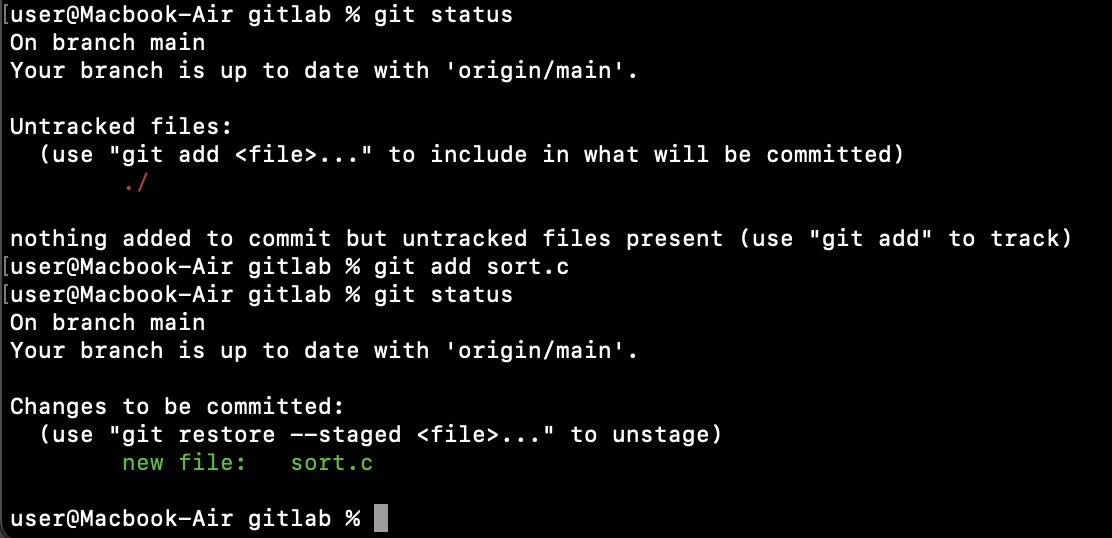
Заходим в репозиторий и создаем папку с лабораторной, вводим git status и git log:



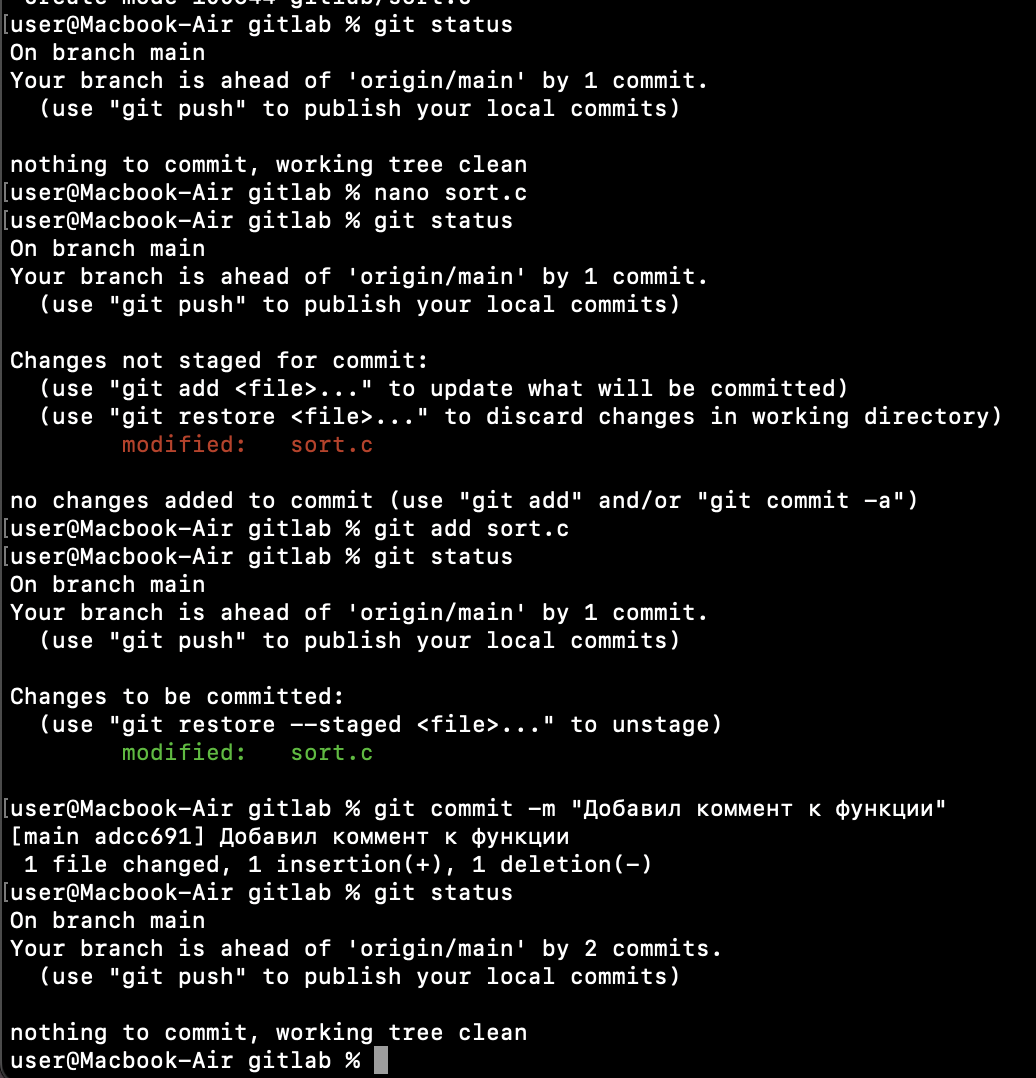
Создаем файл sort.c и добавляем туда функцию СелектСорт:



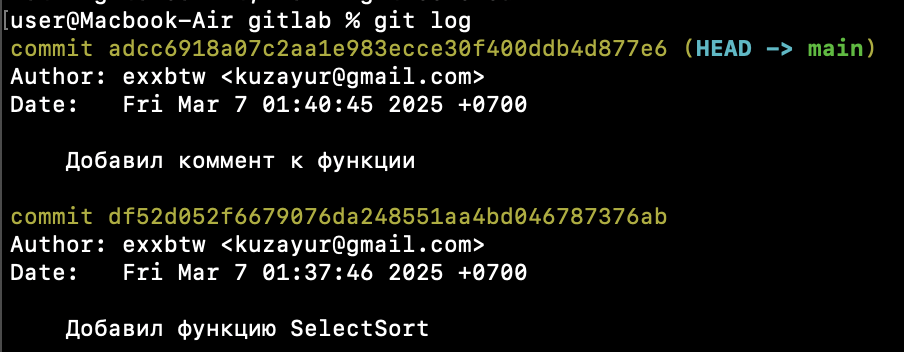
Проверяем изменения через git status, добавляем файл в область stage, и снова вводим git status:



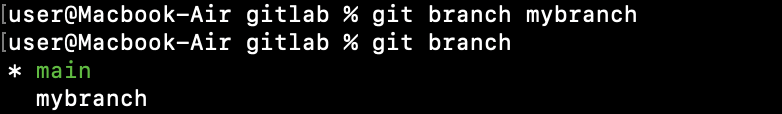
Коммитим файл, и снова прописываем git status:



Вводим git log, чтобы посмотреть, что изменилось в журнале:

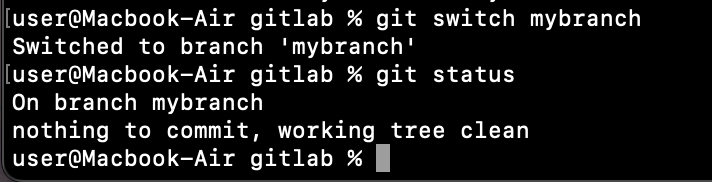


Создаем ветку mybranch, и снова введем git branch, чтобы увидеть новую ветку:

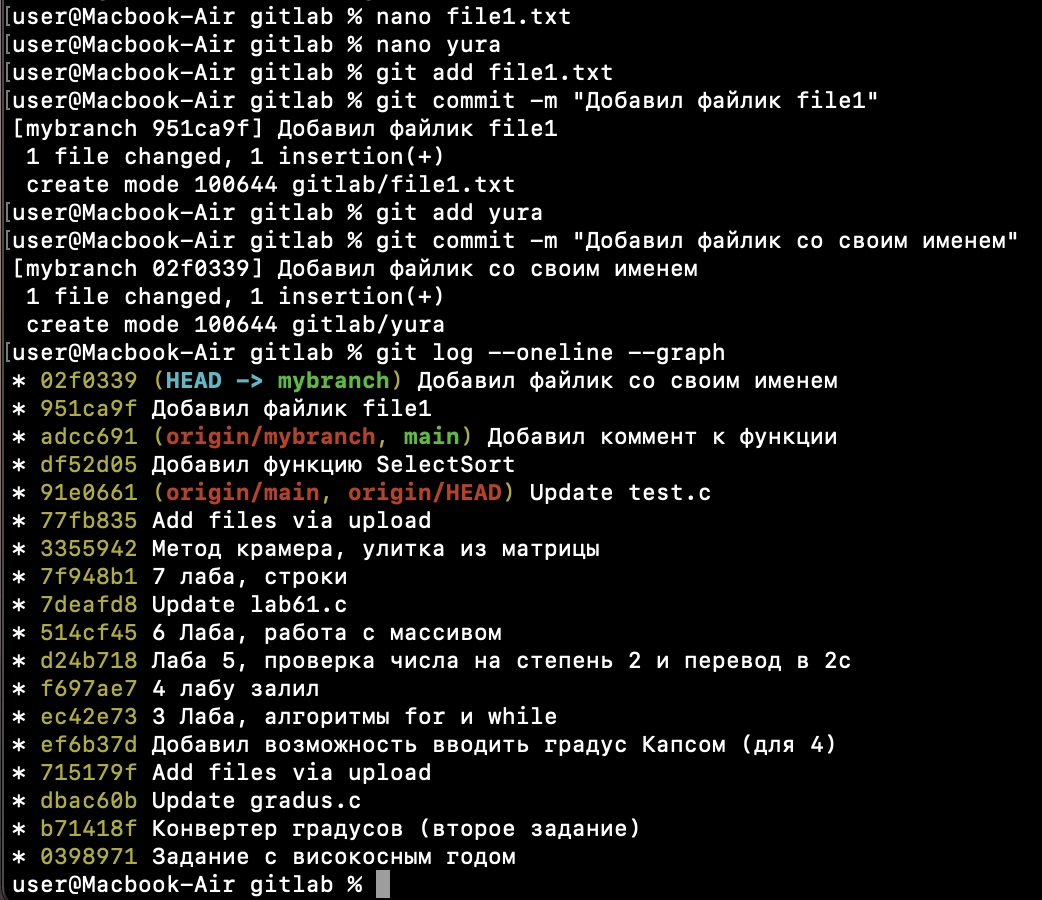


Пушим на ветку, но возникает требование авторизоваться. Авторизуемся с помощью сгенерированного на гитхабе ключа, и успешно пушим на репозиторий:

Переключаемся на новую ветку с помощью git switch, и проверяем результат:



Создаем файл с именем file1.txt и своим именем, сразу же добавляем файлы и коммитим изменения. Используем git log --oneline --graph, чтобы увидеть, что наша ветка указывает на новый коммит:

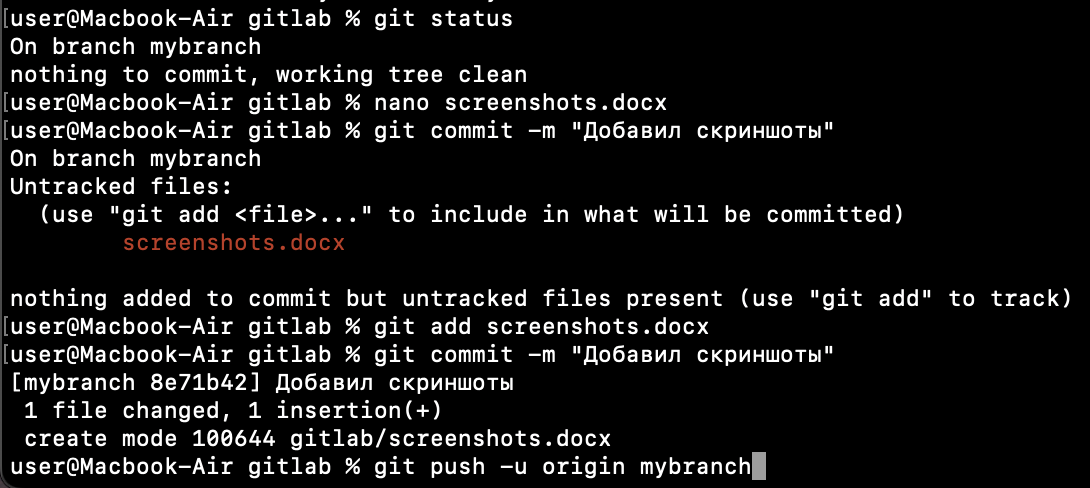


1. Перейдем в ветку main. Тут мы создаем файл file2, добавляем и сразу коммитим. Теперь пропишем git log --oneline --graph --all, чтобы увидеть, что ветка указывает на новый коммит, и что теперь у двух веток разные коммиты:

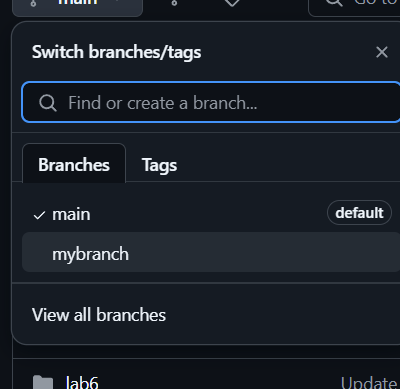


Если перейти в другую ветку, там этого файла не увидим. Нет, он не пропал, он остался в ветке main.

Добавим текстовый документ screenshots в ветку mybranch. Коммитим и пушим на удаленный репо в ветку mybranch:



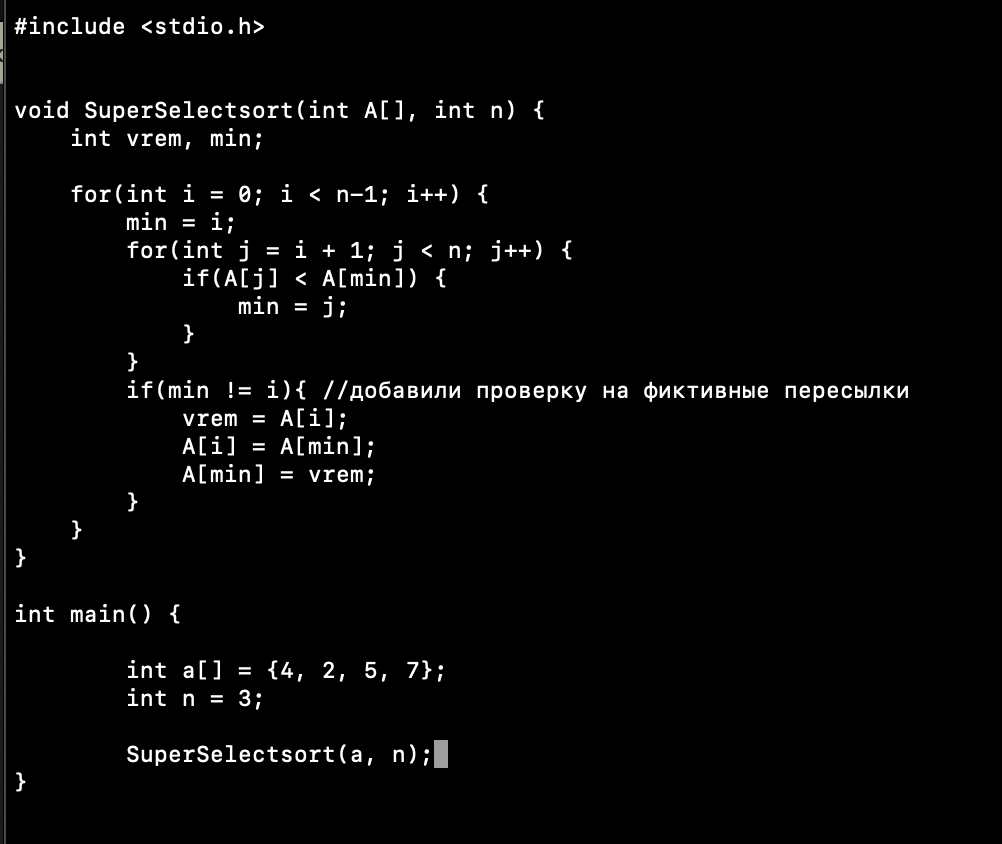
Убеждаемся, что в гитхабе есть 2 ветки:



На четверочку

Переключаемся на ветку mybranch с помощью git switch mybranch

Перезаписываем sort.c , добавив main() и массив из чисел(я решил изменить прошлую функцию, без структуры student):



Git diff выведет следующие строчки:



Git diff –staged Ничего не выдает.

После git add sort.c, diff --staged выдает те же изменения файла, А вот git diff уже становится пустым.

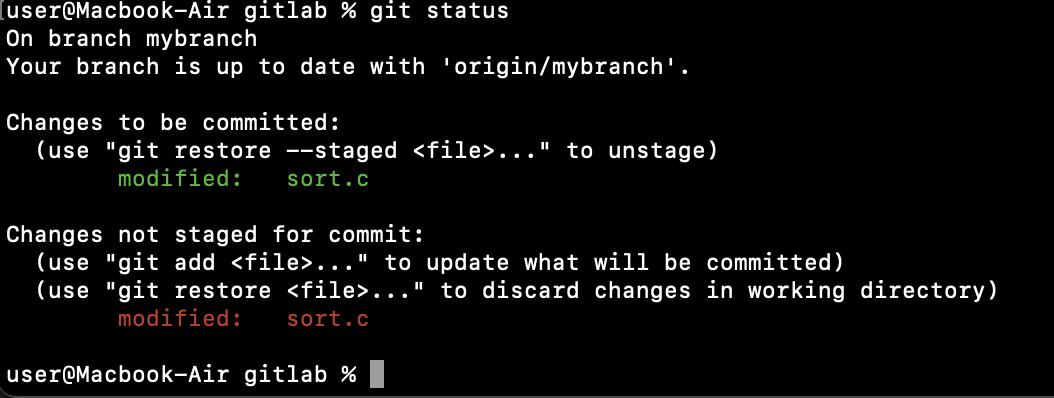
После удаления любого числа из массива git diff указал на это:



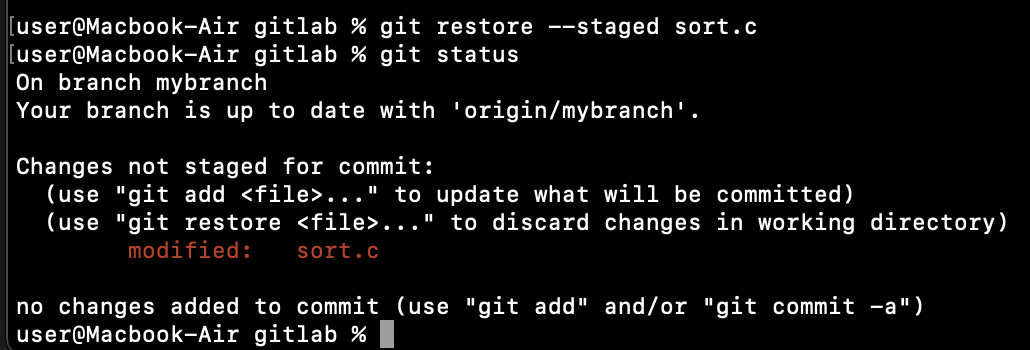
А git diff –staged покажет предыдущие изменения, т.к. Мы еще не сделали git add sort.c.

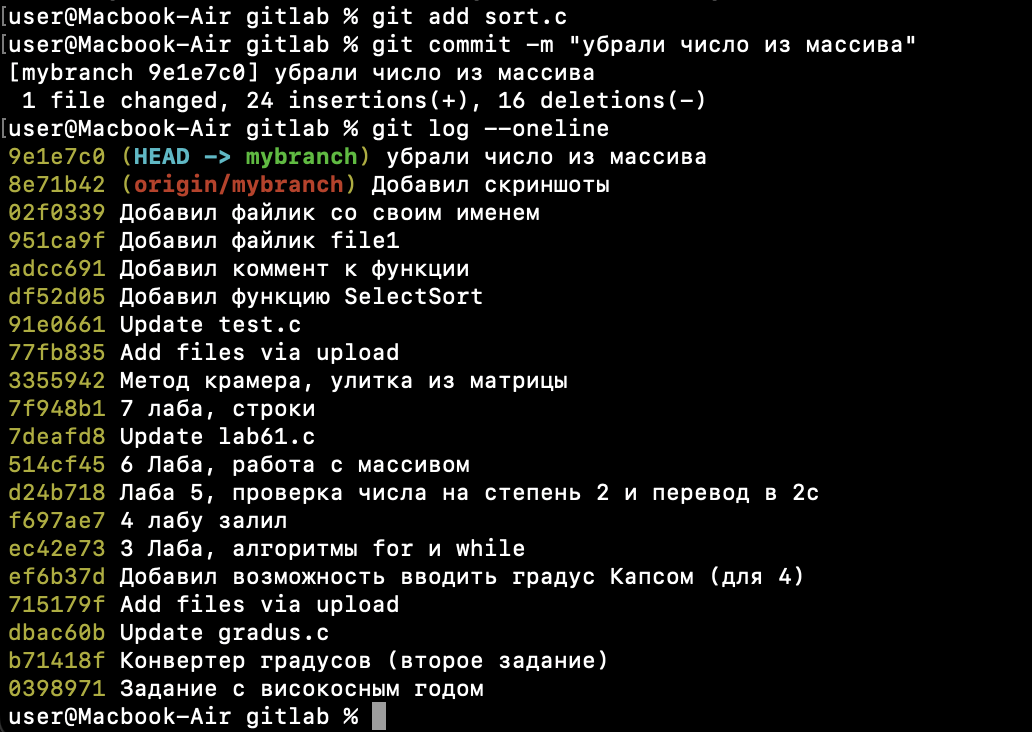
Git diff показывает изменения в рабочей директории, которые еще не добавлены в staging area. А git diff –staged показывает изменения в добавленных файлах

В git status sort.c присутствует дважды в выводе:

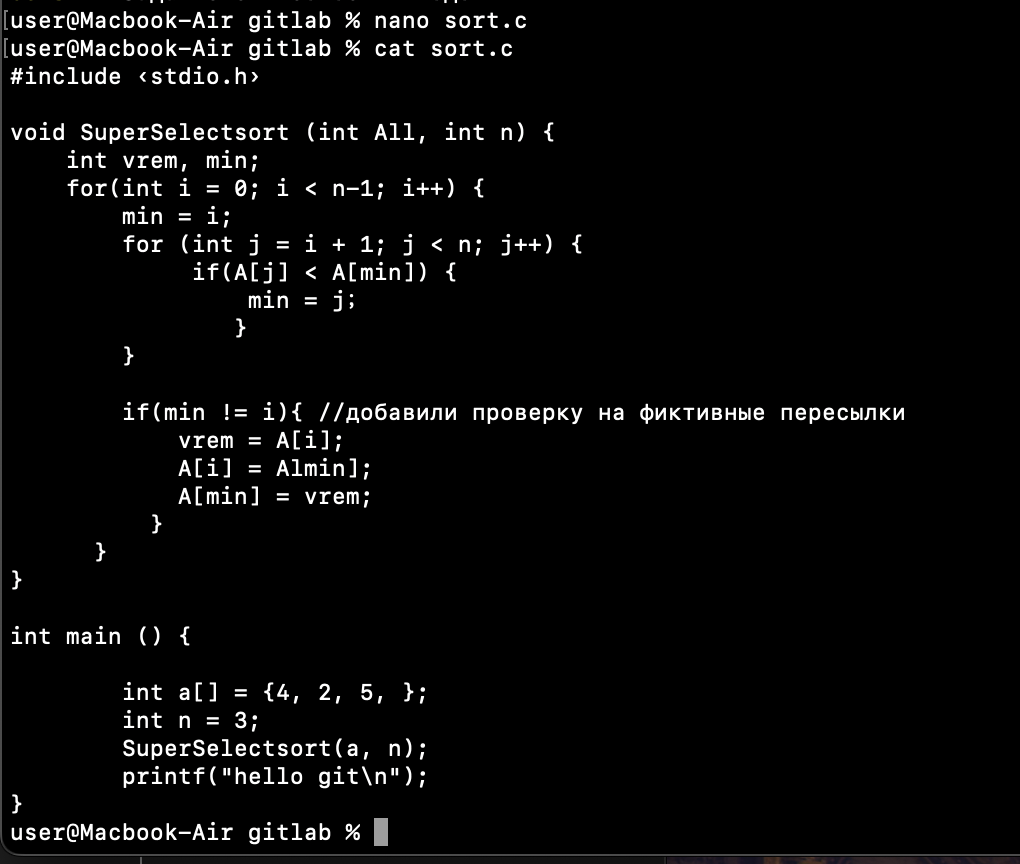


После ввода git restore –staged sort.c(отмена индексации) git status пишет, что sort.c находится только в разделе Changes not staged for commit:

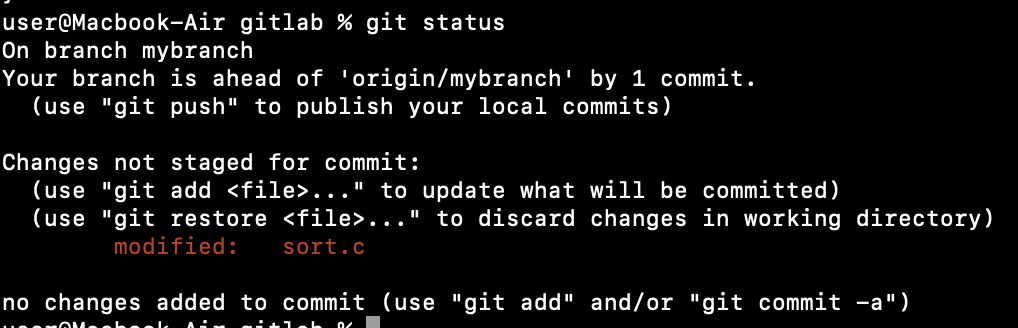


После добавления изменения в staging area(add) git log выглядит так:  


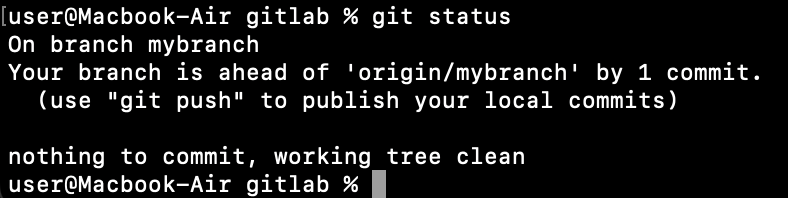
Добавим в sort.c в main() printf(“hello git\n”); и проверим содержимое:



Git status говорит нам, что файл изменен, но изменения не добавлены в staging area:



После ввода git restore sort.c у нас происходит отмена изменений, и строчка printf пропадает. Git status говорит, что все изменения отменены и все чисто:

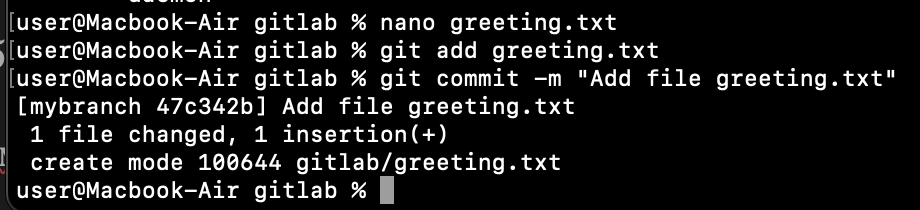


Пушим git push -u origin mybranch.

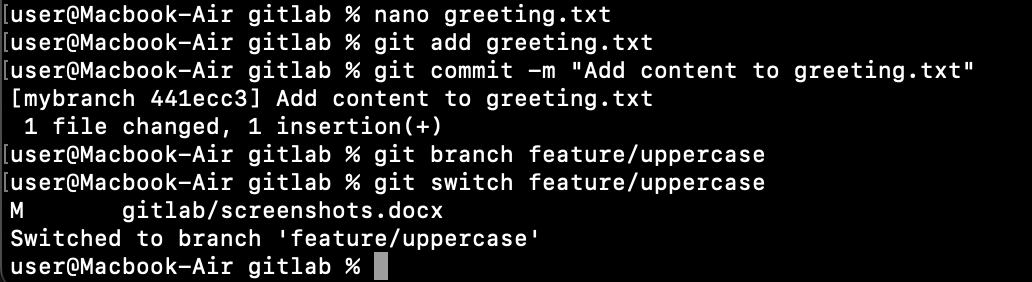
Играемся с ветками и ff merge

когда вы хотите обновить ветку так, чтобы она также имела все изменения из другой ветки, используйте команду 'git merge [имя ветки]', где [имя ветки] - ветка, из которой мы хотим смержить наши изменения.

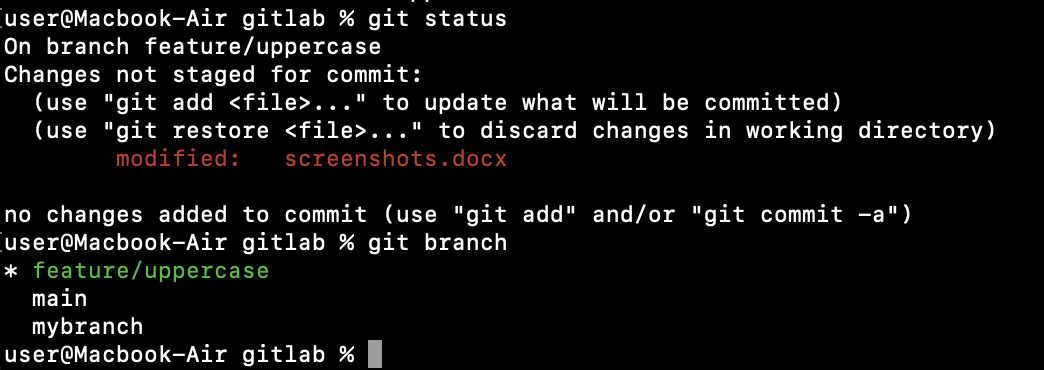
Создаем greeting.txt, добавляем в область и коммитим:



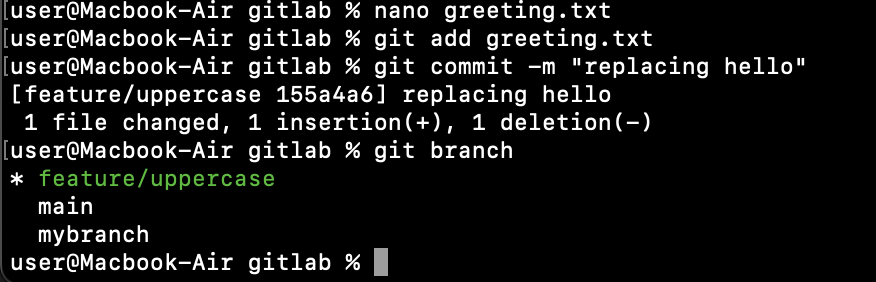
Добавим Hello в greeting.txt, также добавим в область и закоммитим. Затем создаем ветку feature/uppercase и заходим в нее:



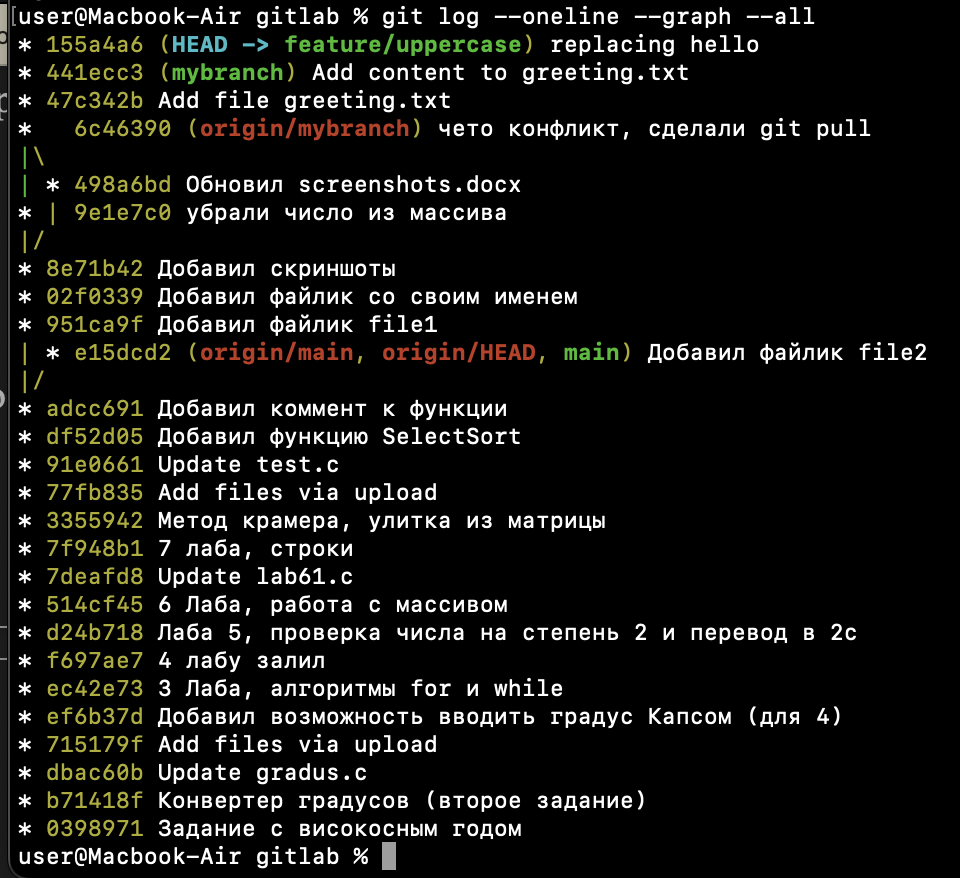
Git status пишет: no changes added to commit:

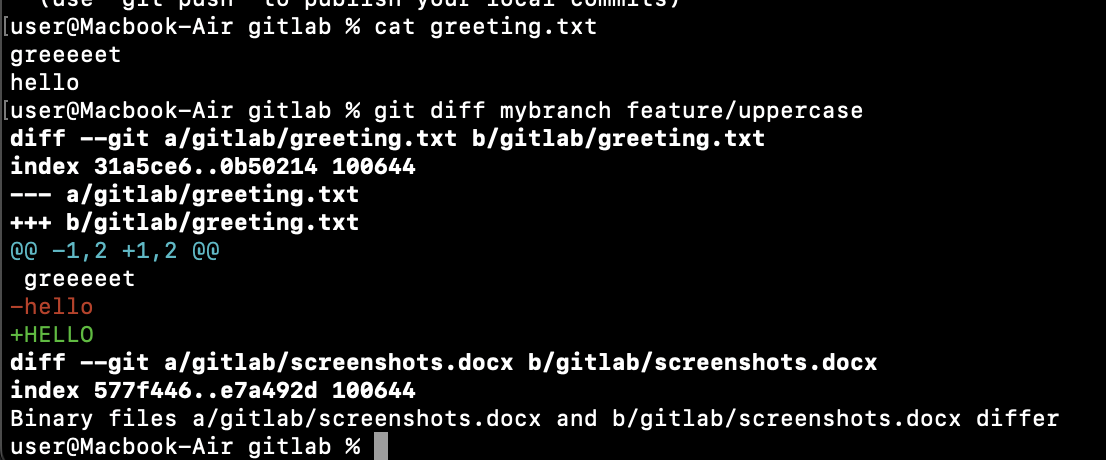


Заменяем hello на HELLO, добавляем и коммитим. Проверяем ветку:

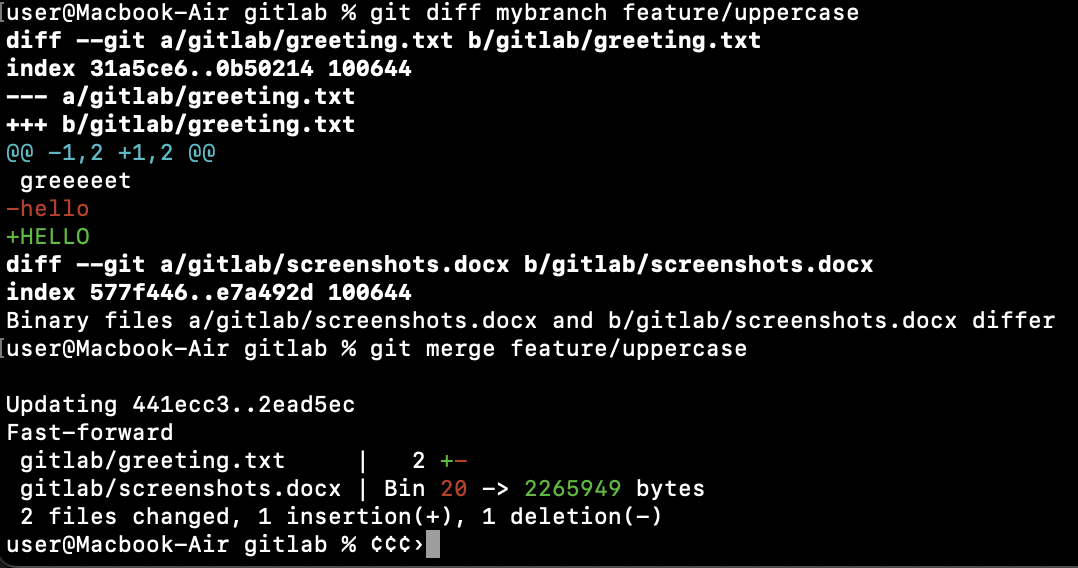


git log --oneline --graph –all выводит следующие строки:

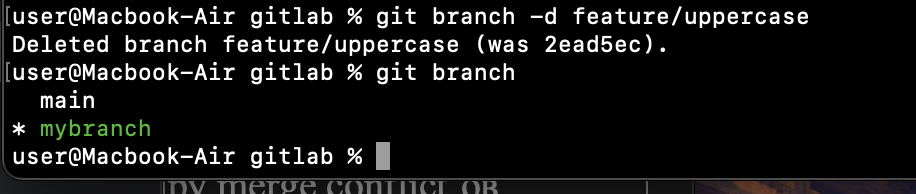


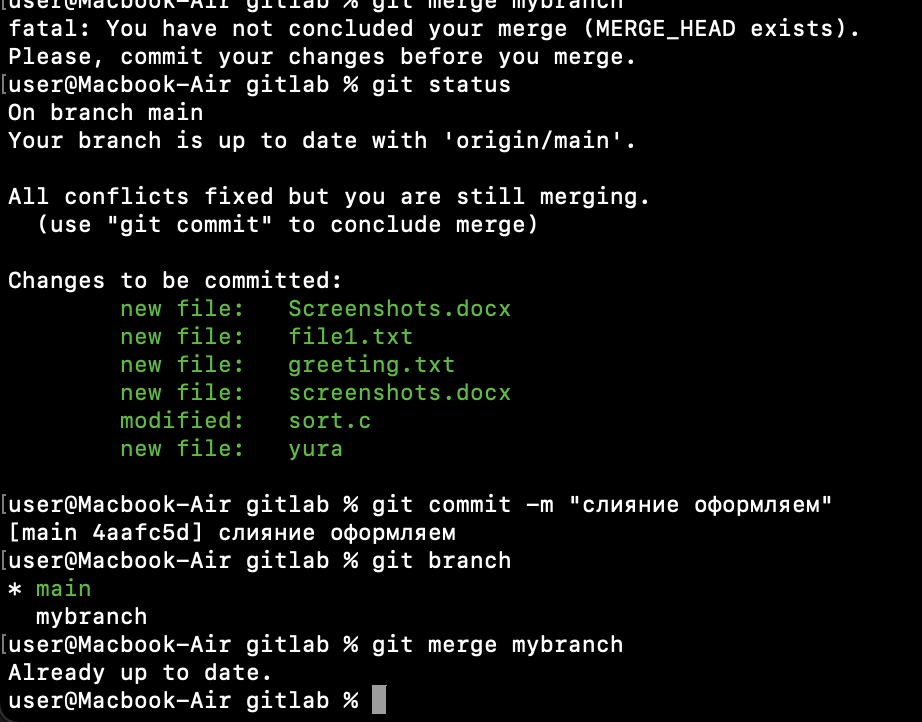
Переходим на ветку mybranch. Вводим cat greeting.txt. А затем сравниваем ветки:  


Объединяем mybranch и feature/uppercase с помощью merge:



Просматриваем greeting.txt и видим там HELLO. Удаляем ветку feature/uppercase и проверяем результат:

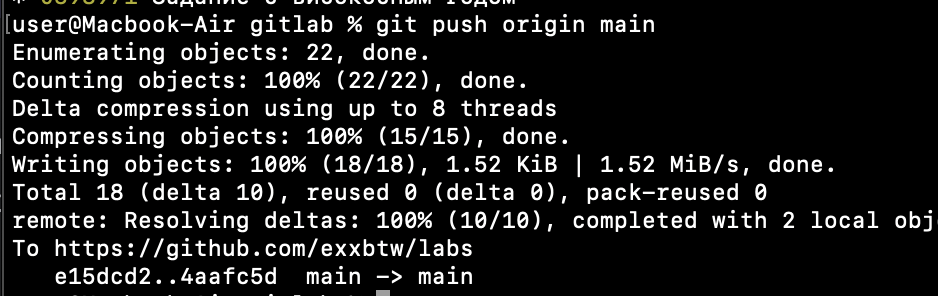


Делаем слияние main и mybranch, решаем проблемы: 

Смотрим git log –oneline –graph --all



Пушим изменения main на репозиторий:



Теперь запушим этот файл на репо, первоначально скопировав его в папку gitlab, а затем добавив и запушив на репо.