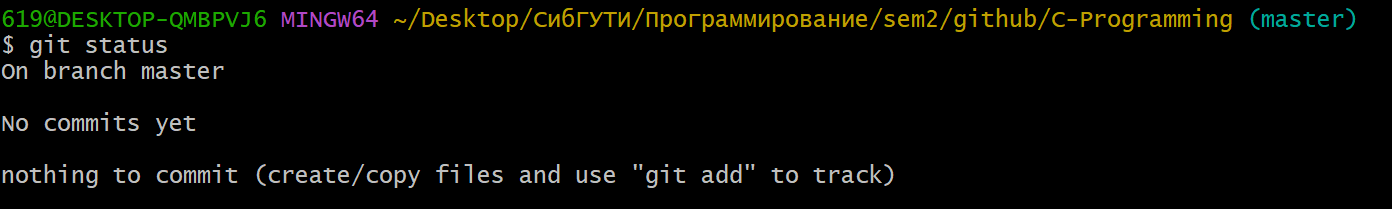
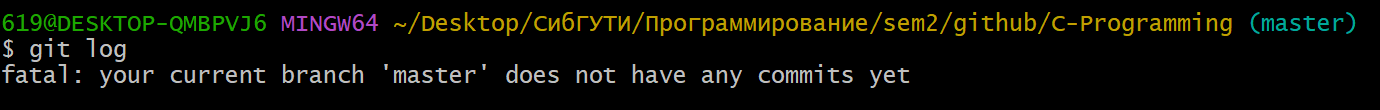
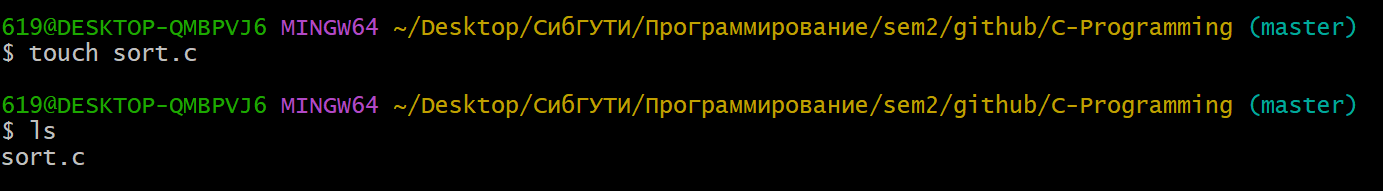
1. Используйте git status, чтобы узнать, на какой ветке вы находитесь.

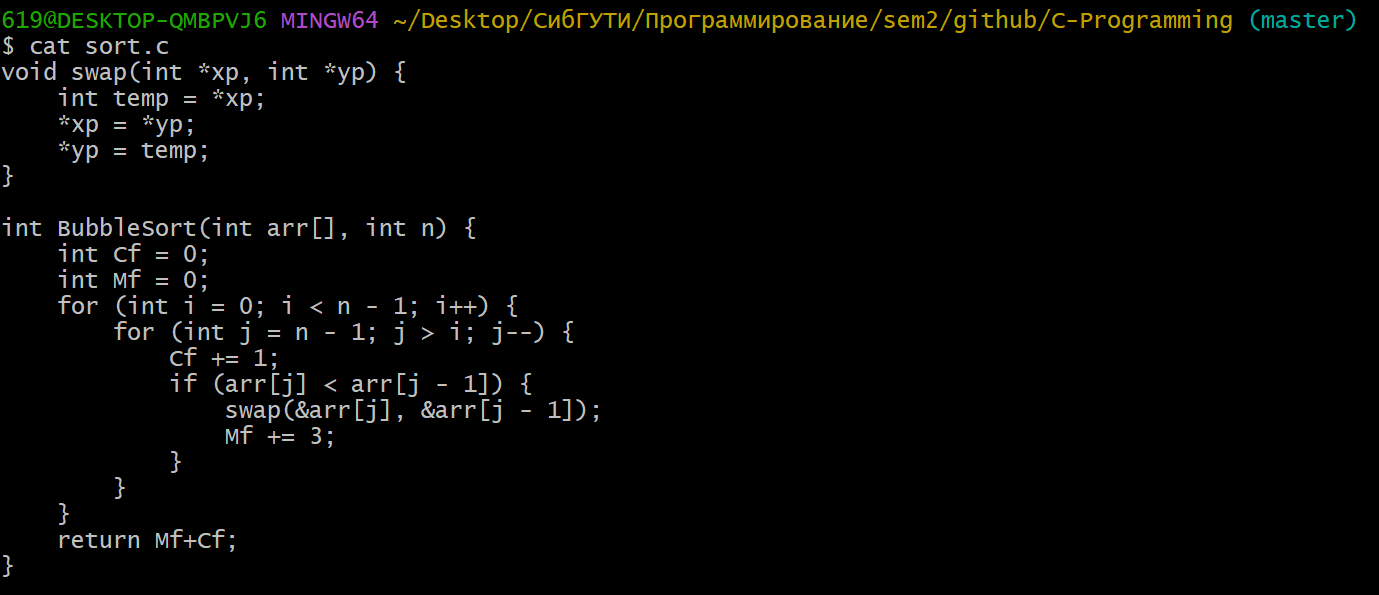


1. Как выглядит git log?

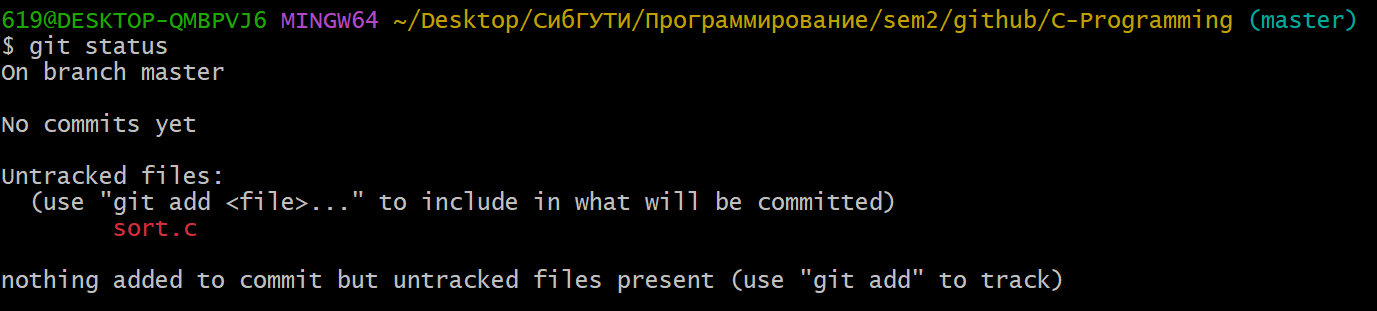


1. Создайте файл sort.c и вставьте туда код функции любой сортировки (**только ф-ия сортировки**)

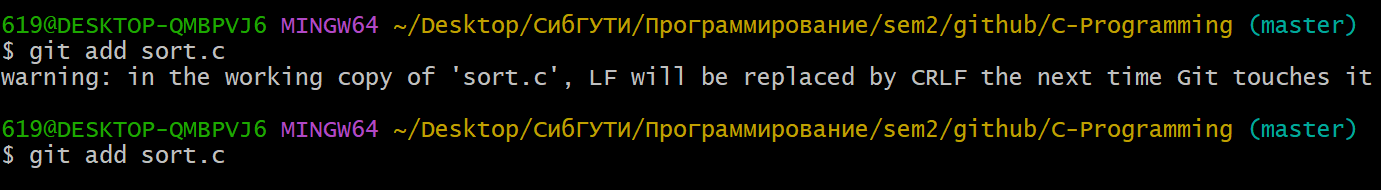


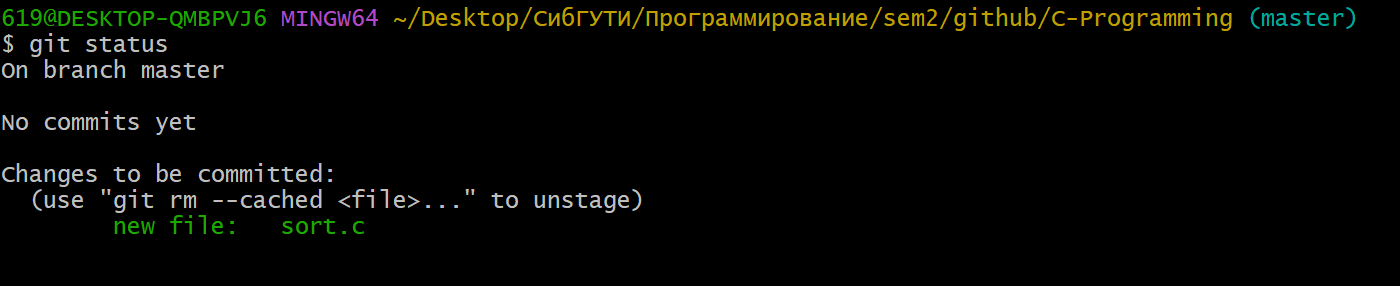
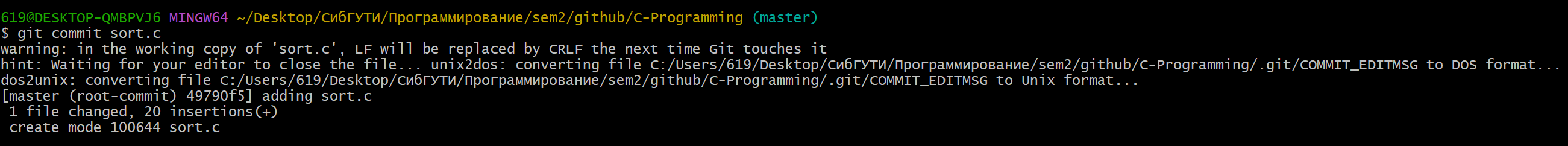
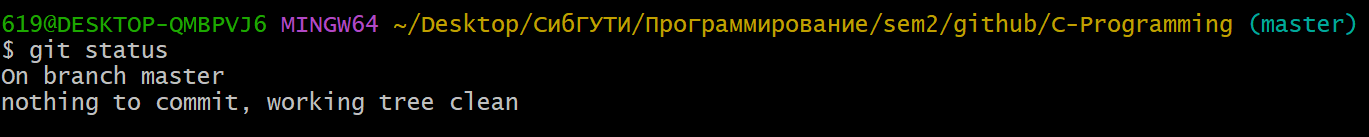
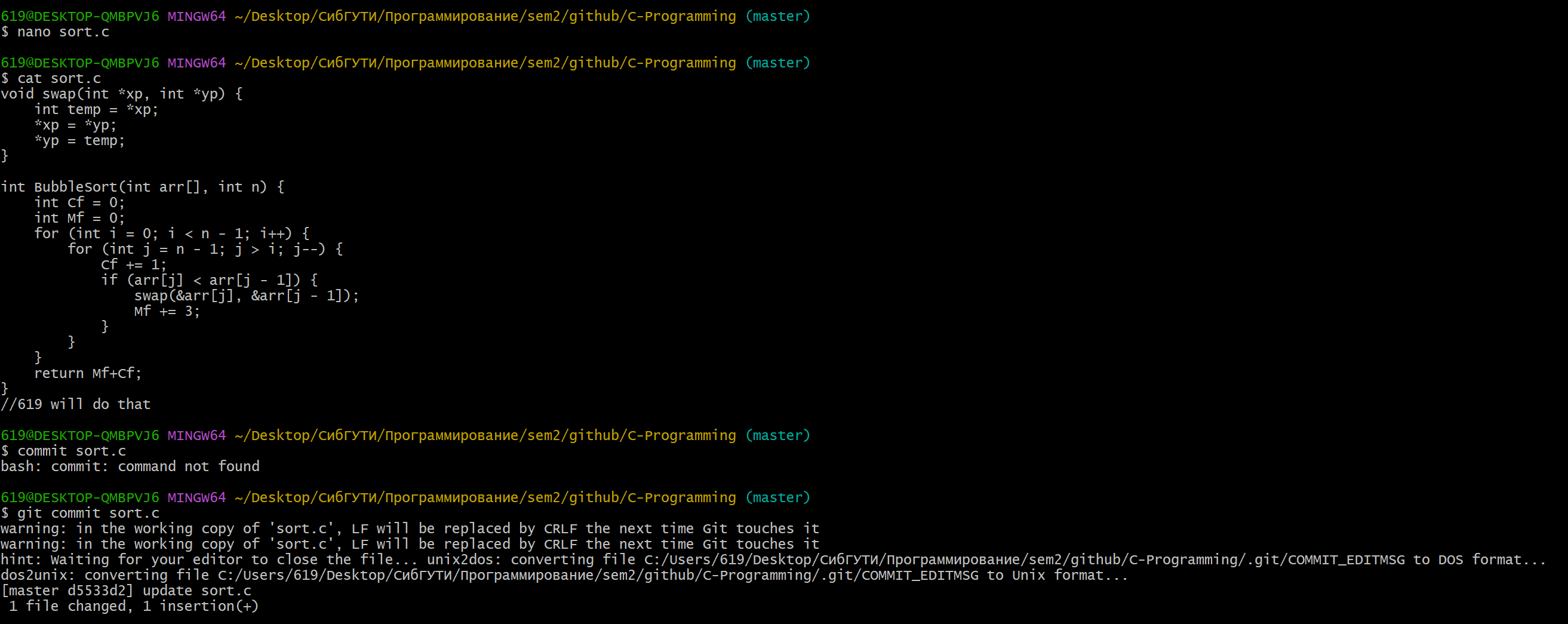
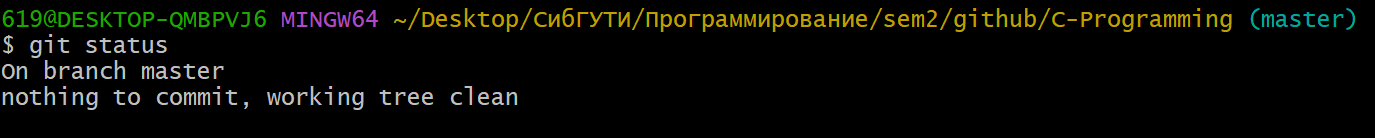


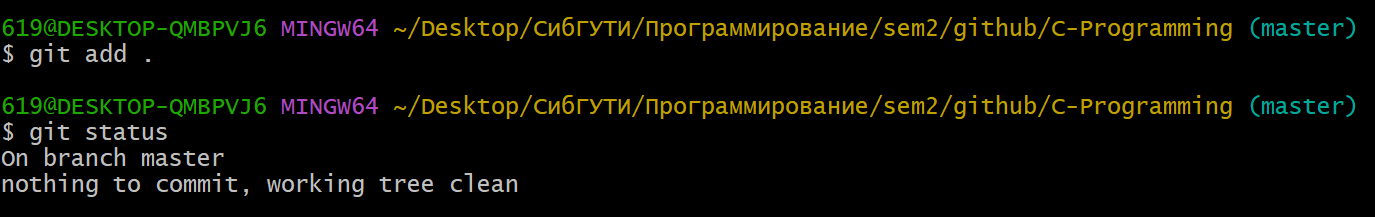
1. Как сейчас выглядит вывод git status?



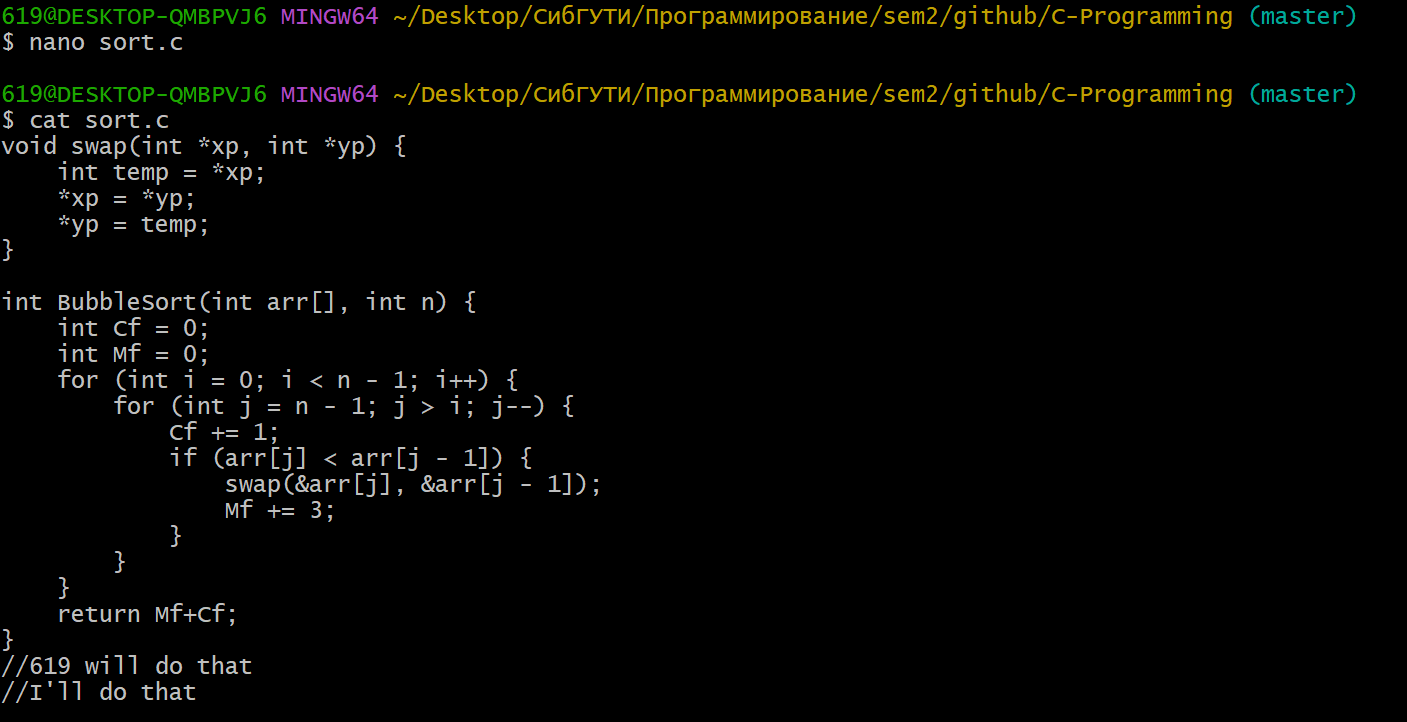
1. Добавьте файл в область stage (add)

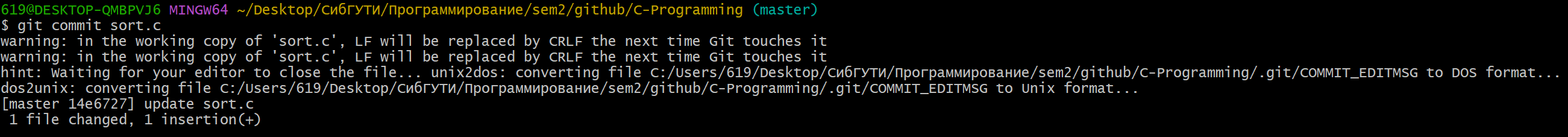
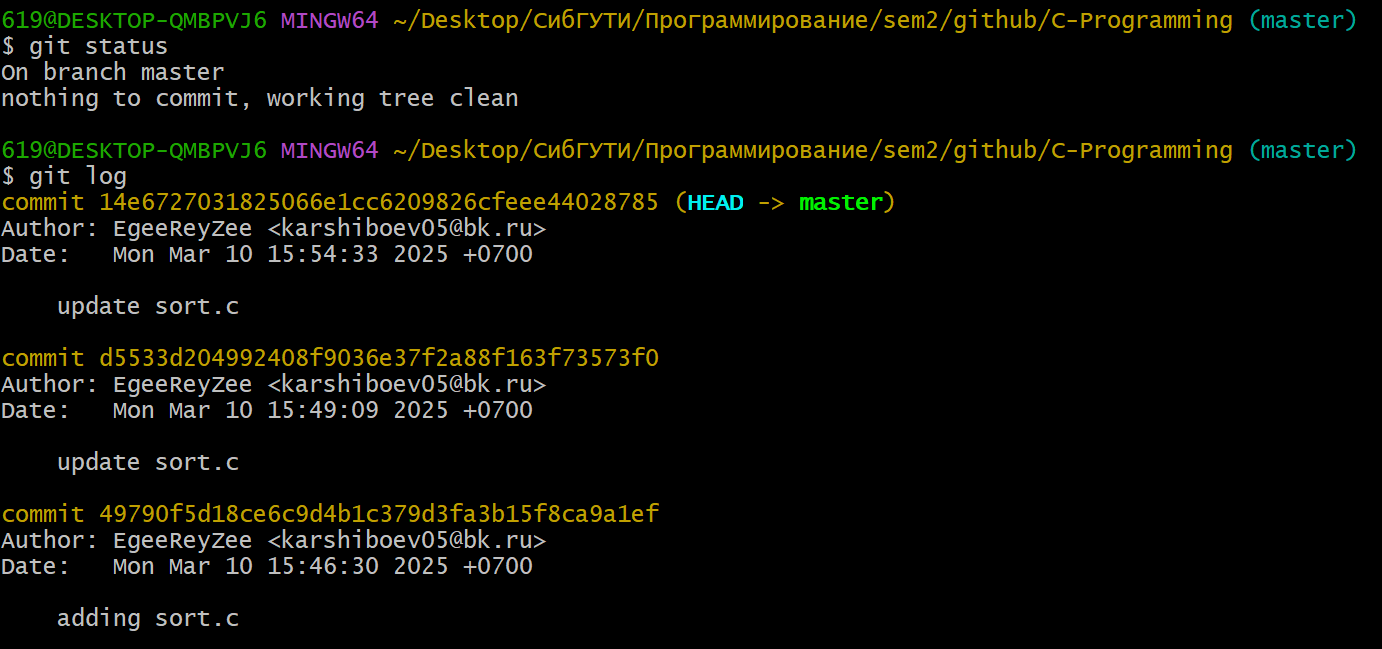
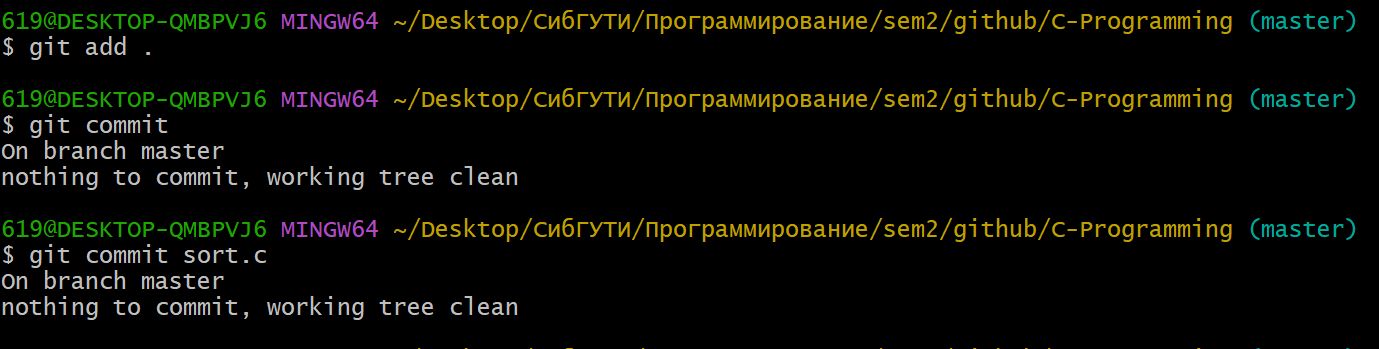
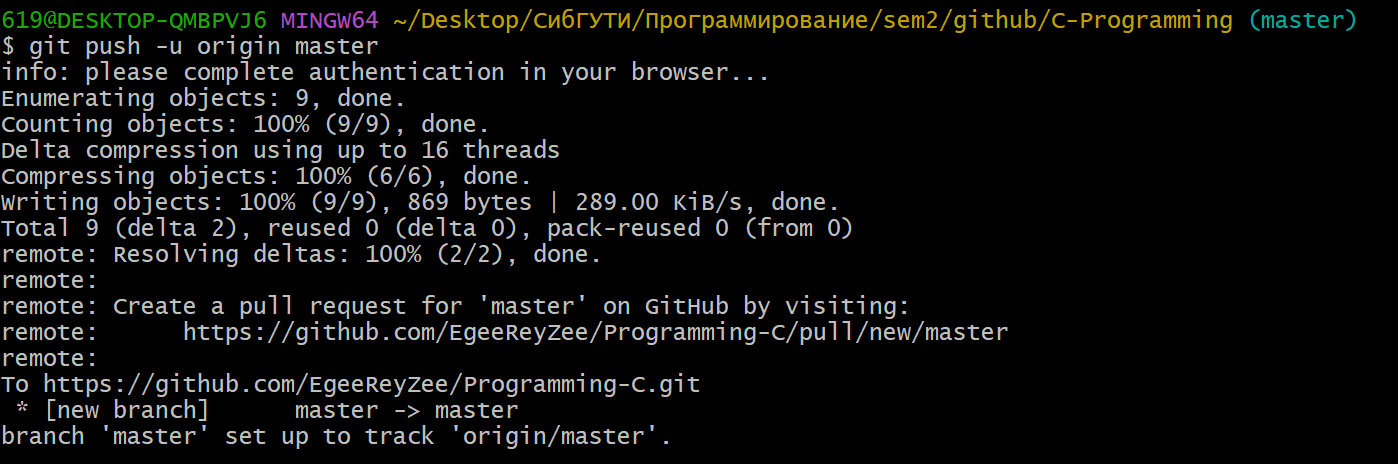


1. Как сейчас выглядит git status? 
2. Закоммитить файл в репозиторий
3. Как сейчас выглядит git status? 
4. Добавить комментарий с любым текстом в этот же файл
5. Как сейчас выглядит git status? 
6. 12. Добавьте (add) изменение файла



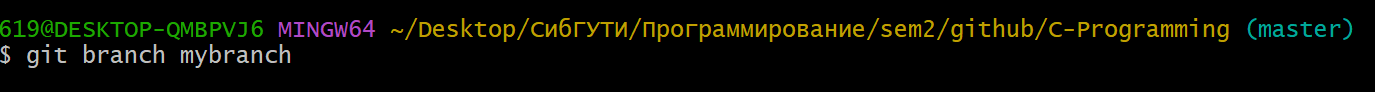
13. Измените файл еще раз (можно добавить еще комментарий или убрать старый)

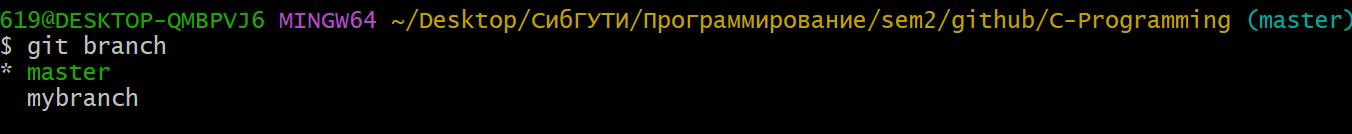
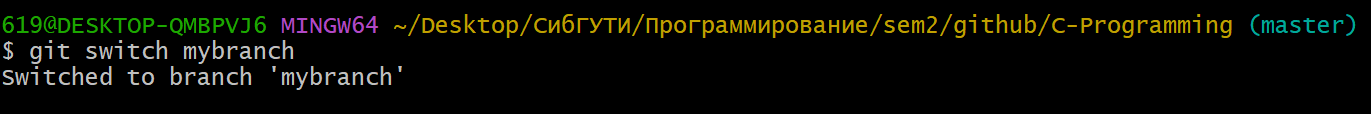
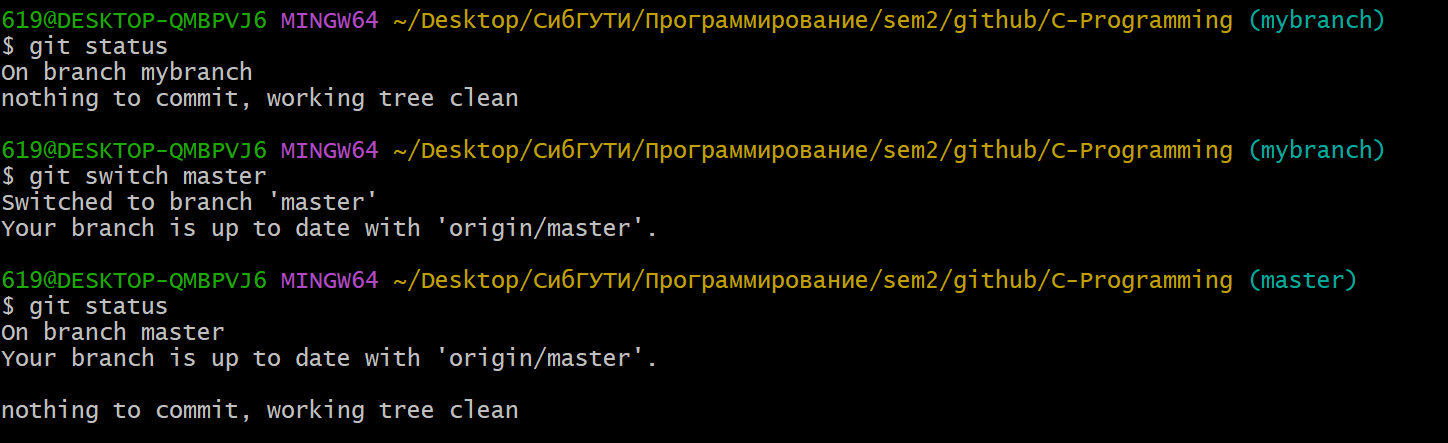
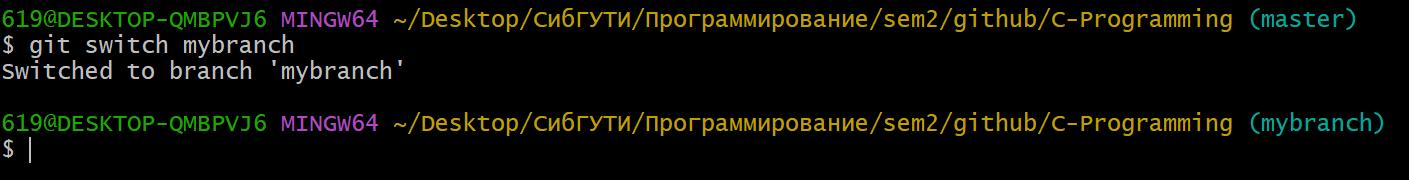
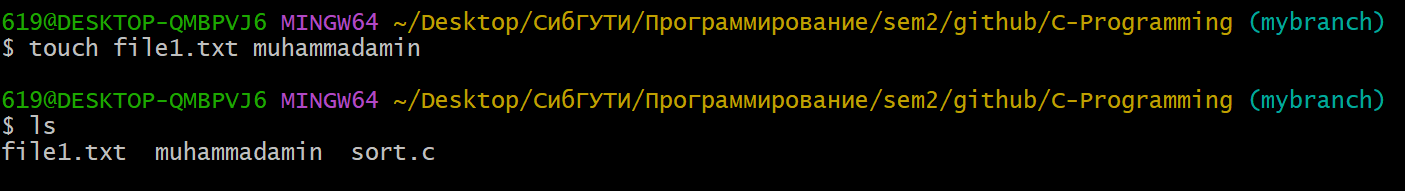
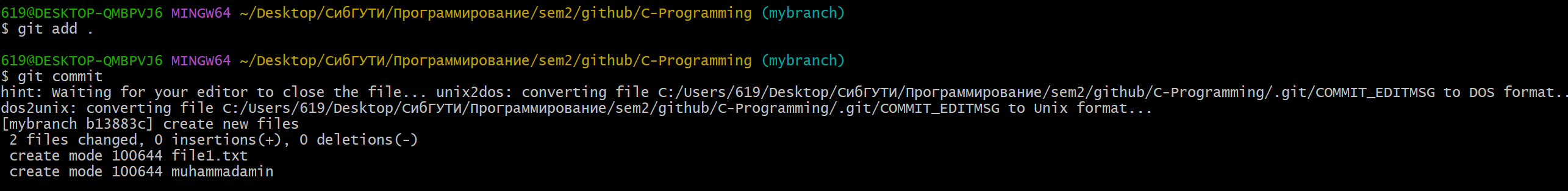
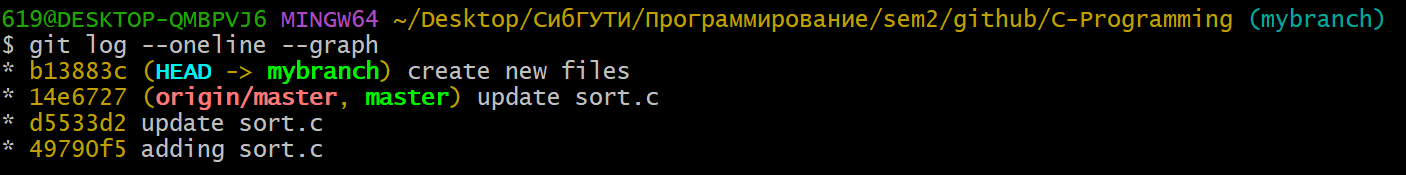
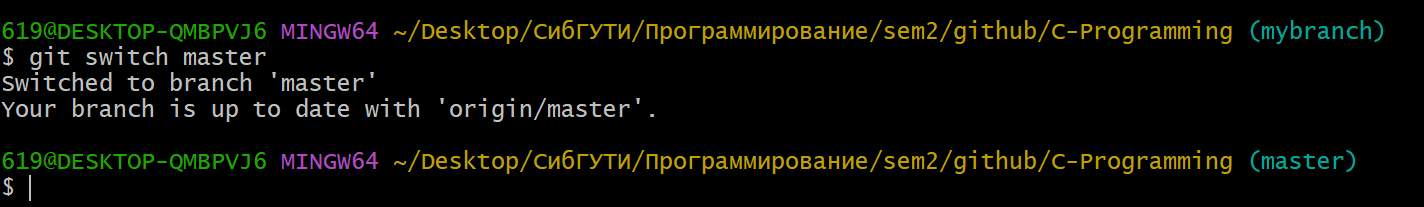
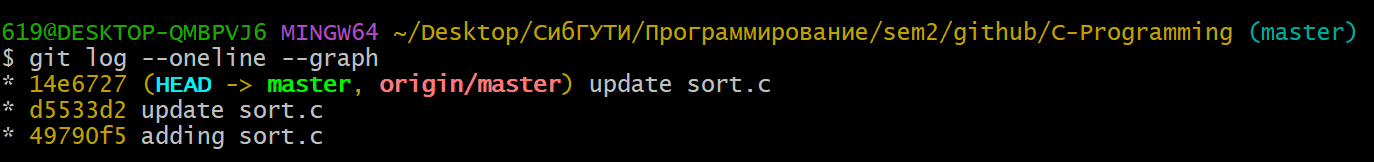
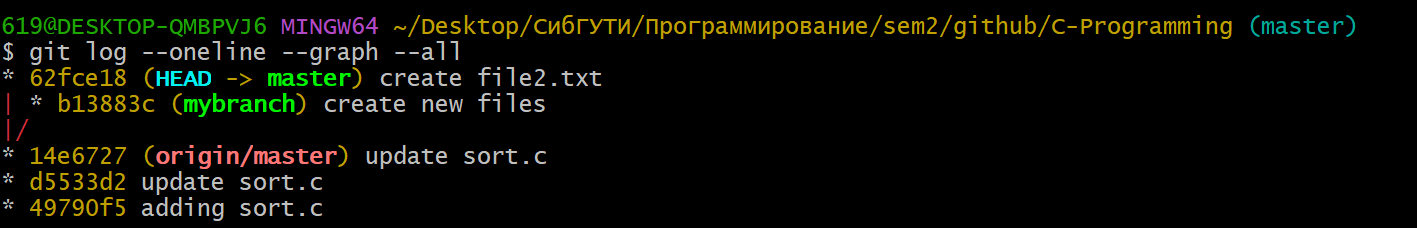
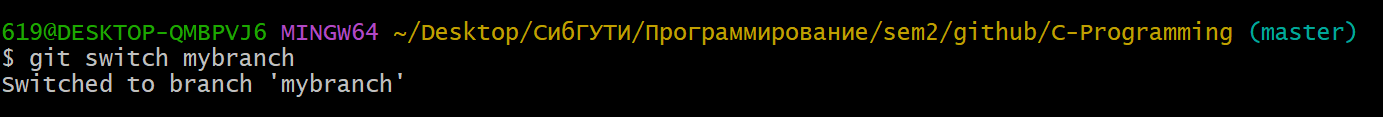
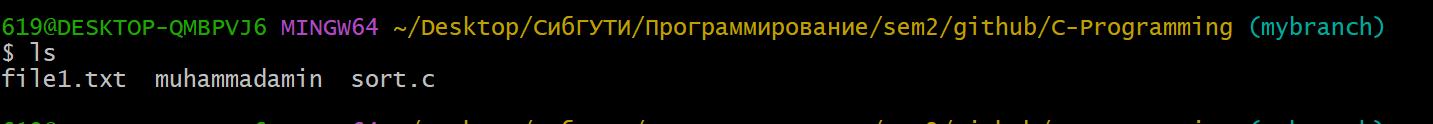
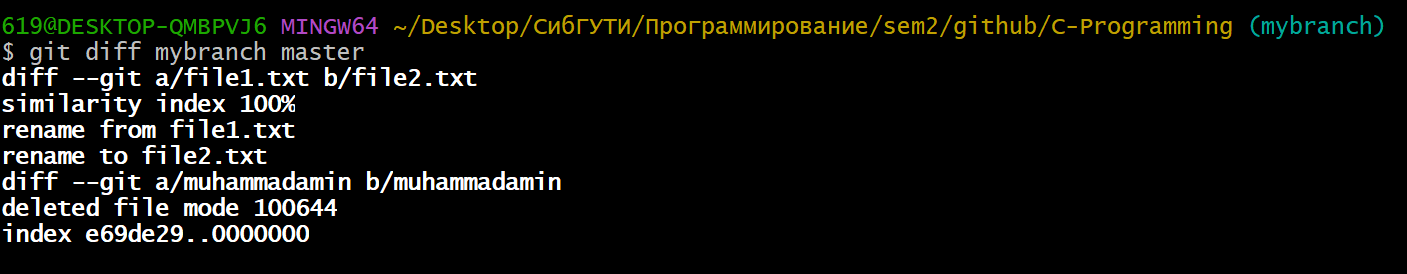
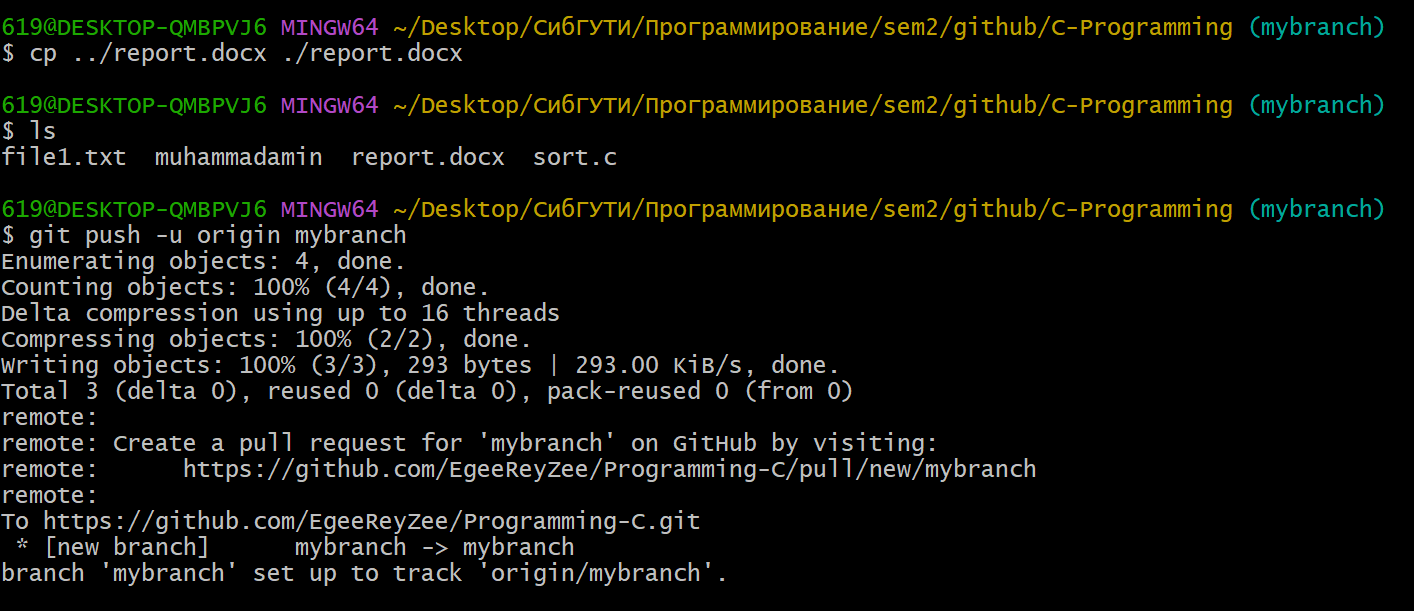
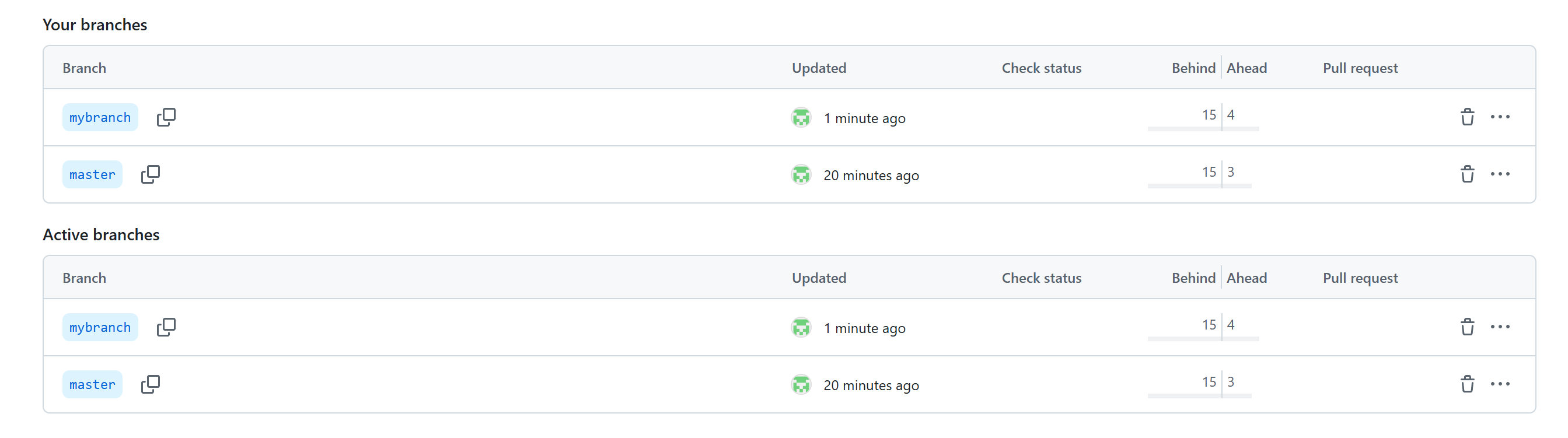


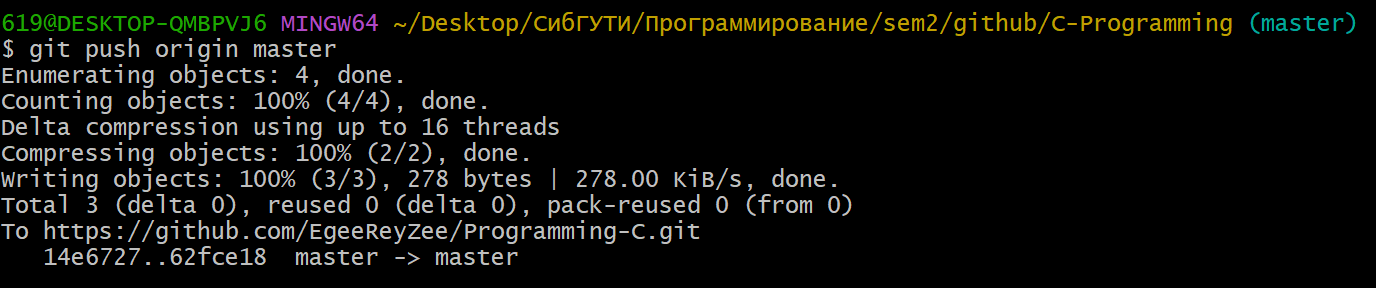
1. Сделайте коммит
2. Как сейчас выглядит status? Журнал (log)? 
3. Добавьте в stage и закоммитьте последнее изменение
4. Запушим на удаленный репо (git push) 

Теперь мы немного поиграем с ветками.

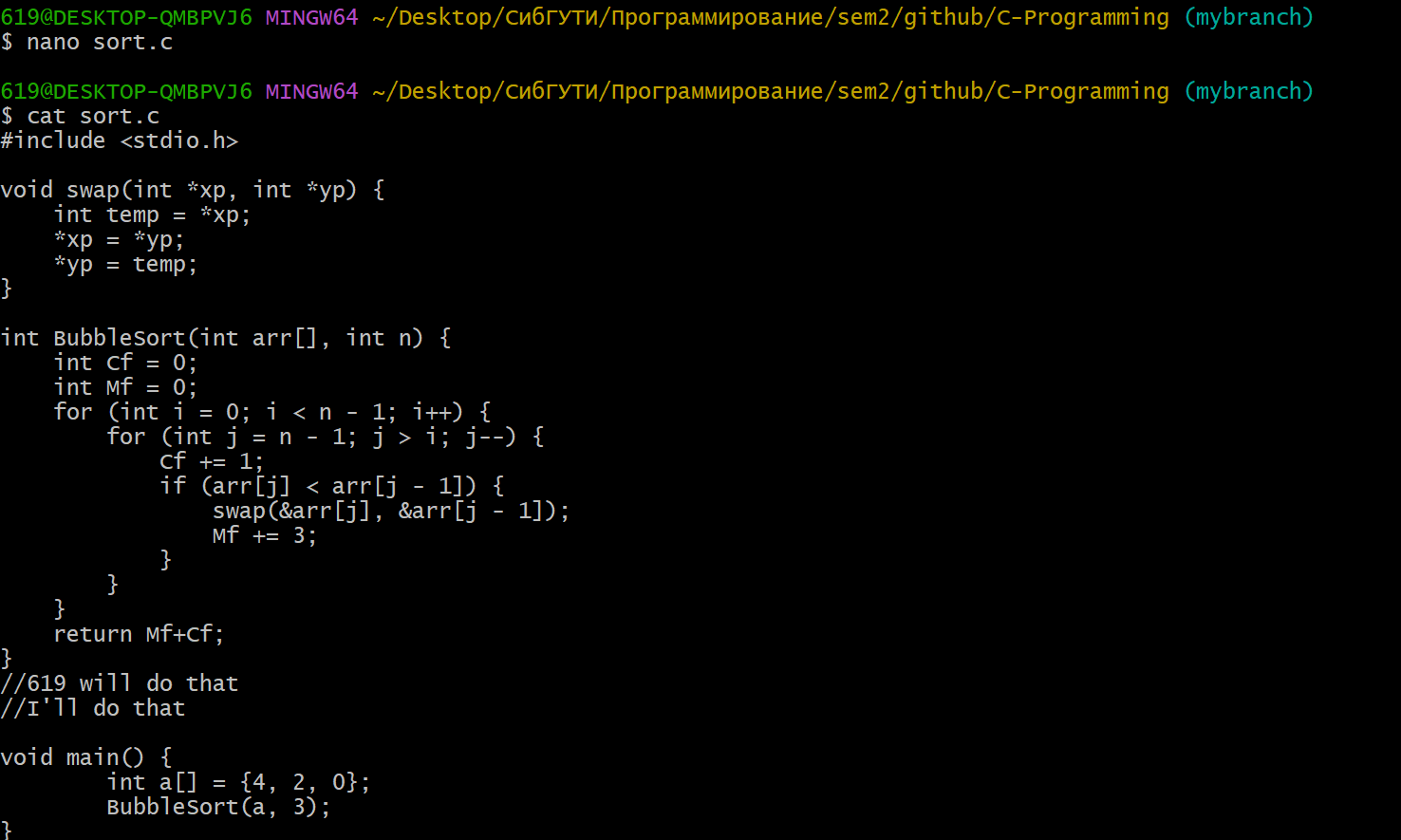
1. Используйте git branch mybranch (или git checkout -b mybranch), чтобы создать новую ветку с именем mybranch.



1. Снова используйте git branch, чтобы увидеть новую созданную ветку. 
2. Используйте git switch mybranch (или git checkout mybranch), чтобы переключиться на новую ветку. 
3. Как изменяется вывод git status при переключении между master и новой веткой, которую вы создали? 
4. Убедитесь, что вы находитесь на своей ветке mybranch, прежде чем продолжить. 
5. Создайте файл с именем file1.txt и своим именем. 
6. Добавьте файл и закоммитьте это изменение. 
7. Используйте git log --oneline --graph, чтобы увидеть, что ваша ветка указывает на новый коммит. 
8. Вернитесь к ветке с именем master. 
9. Используйте git log --oneline --graph, что изменилось? 
10. Создайте новый файл с именем file2.txt и закоммитьте его. 
11. Используйте git log --oneline --graph --all, чтобы увидеть, что ваша ветка указывает на новый коммит, и что теперь у двух веток разные коммиты 
12. Переключитесь на вашу ветку mybranch. 
13. Наш file2.txt пропал? 
14. Используйте git diff mybranch master, чтобы увидеть разницу между двумя ветками. 
15. Добавить текстовый документ со скриншотами в ветку **mybranch**. Закоммитить и запушить на удаленный репо ветку mybranch (git push -u origin mybranch) 
16. Убедиться что в github.com две ветки master и mybranch. Не забыть запушить изменения master ветки в master 

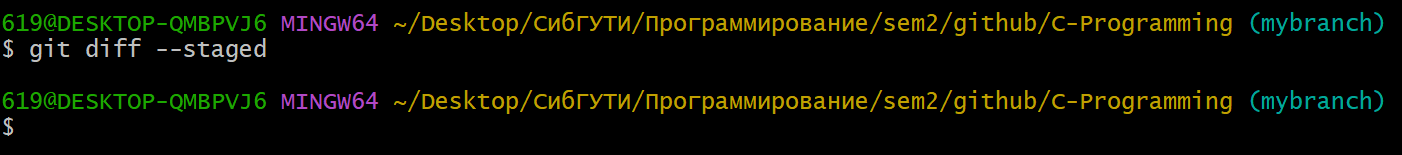
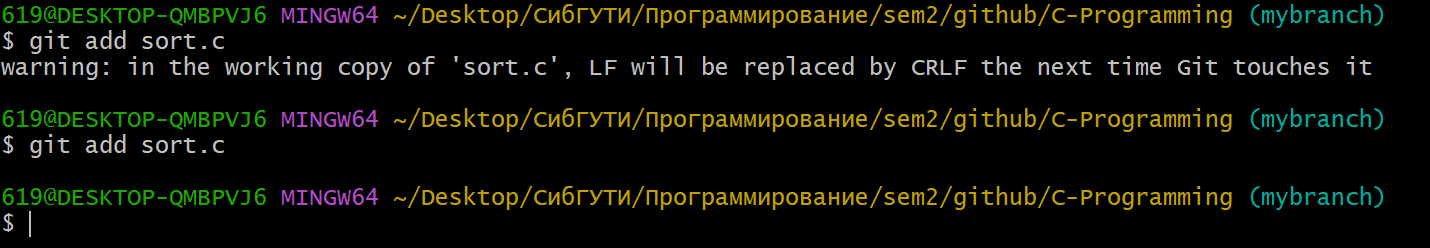
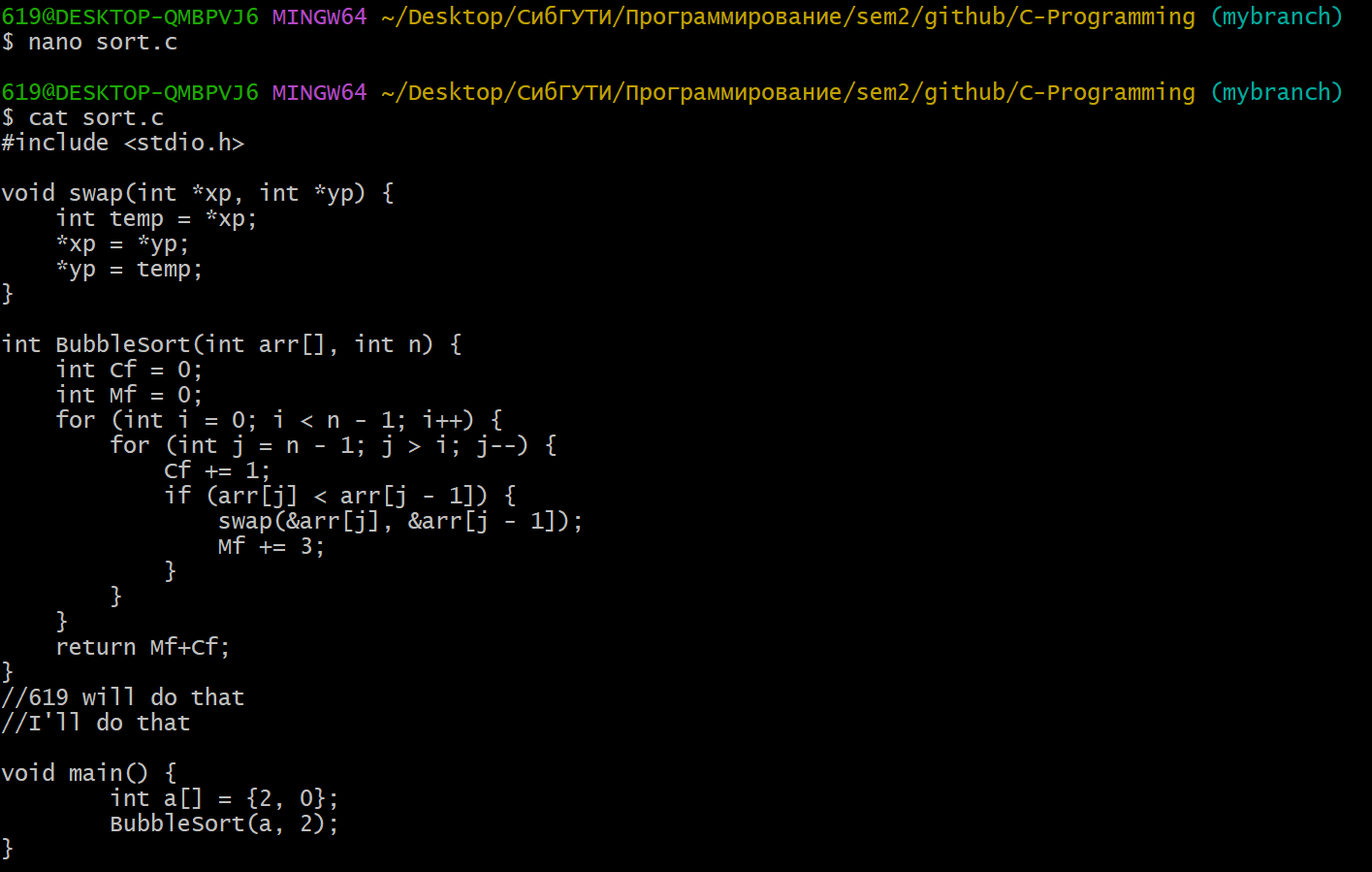
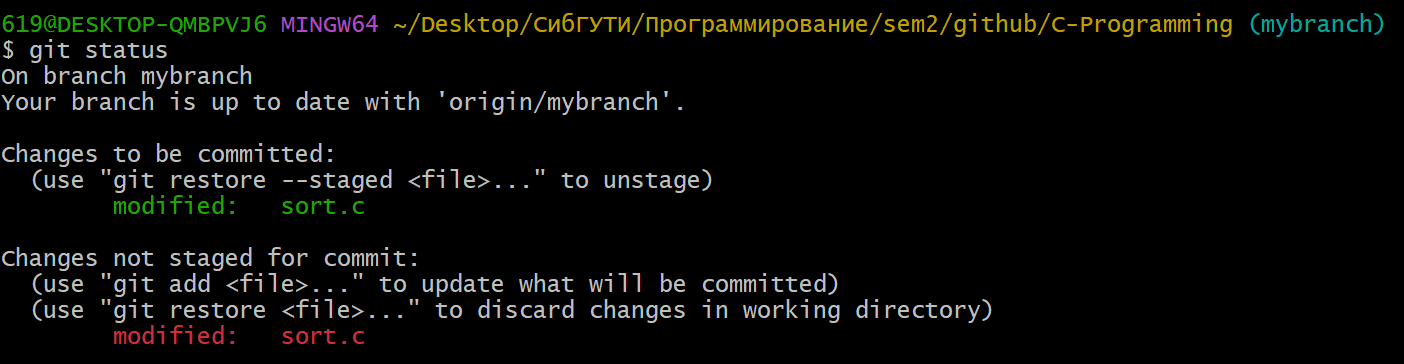
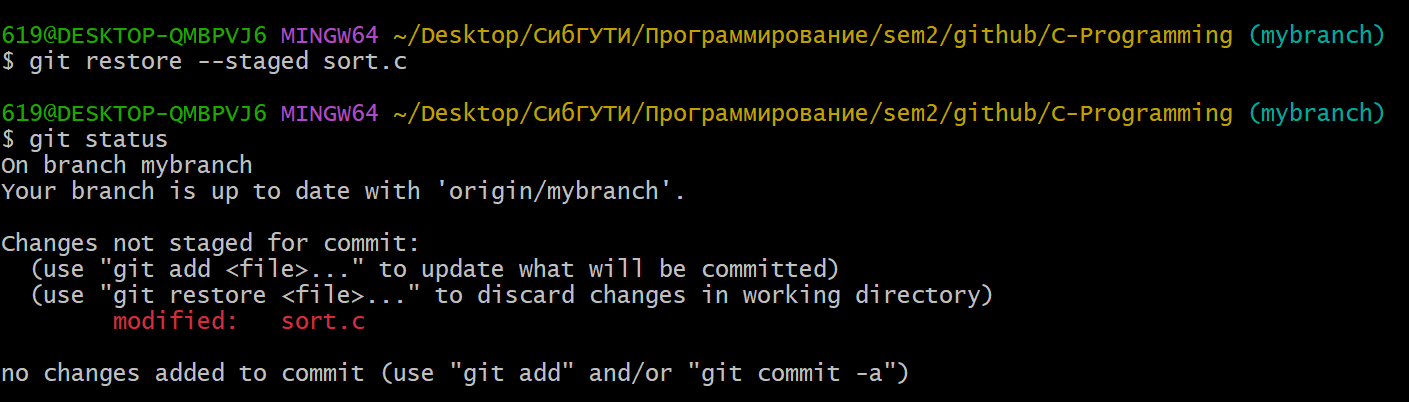
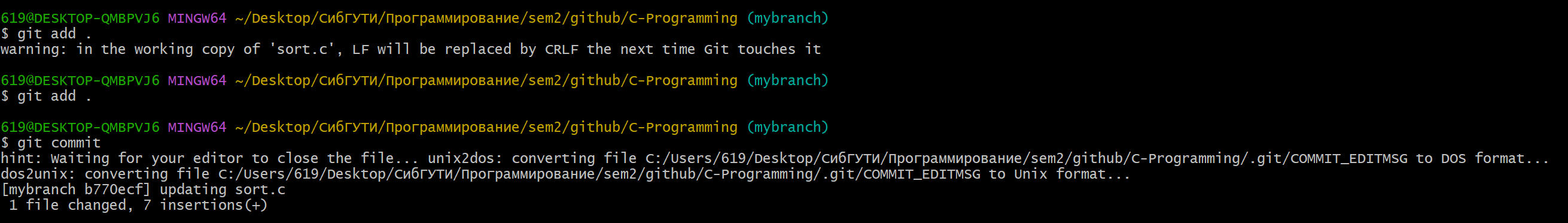
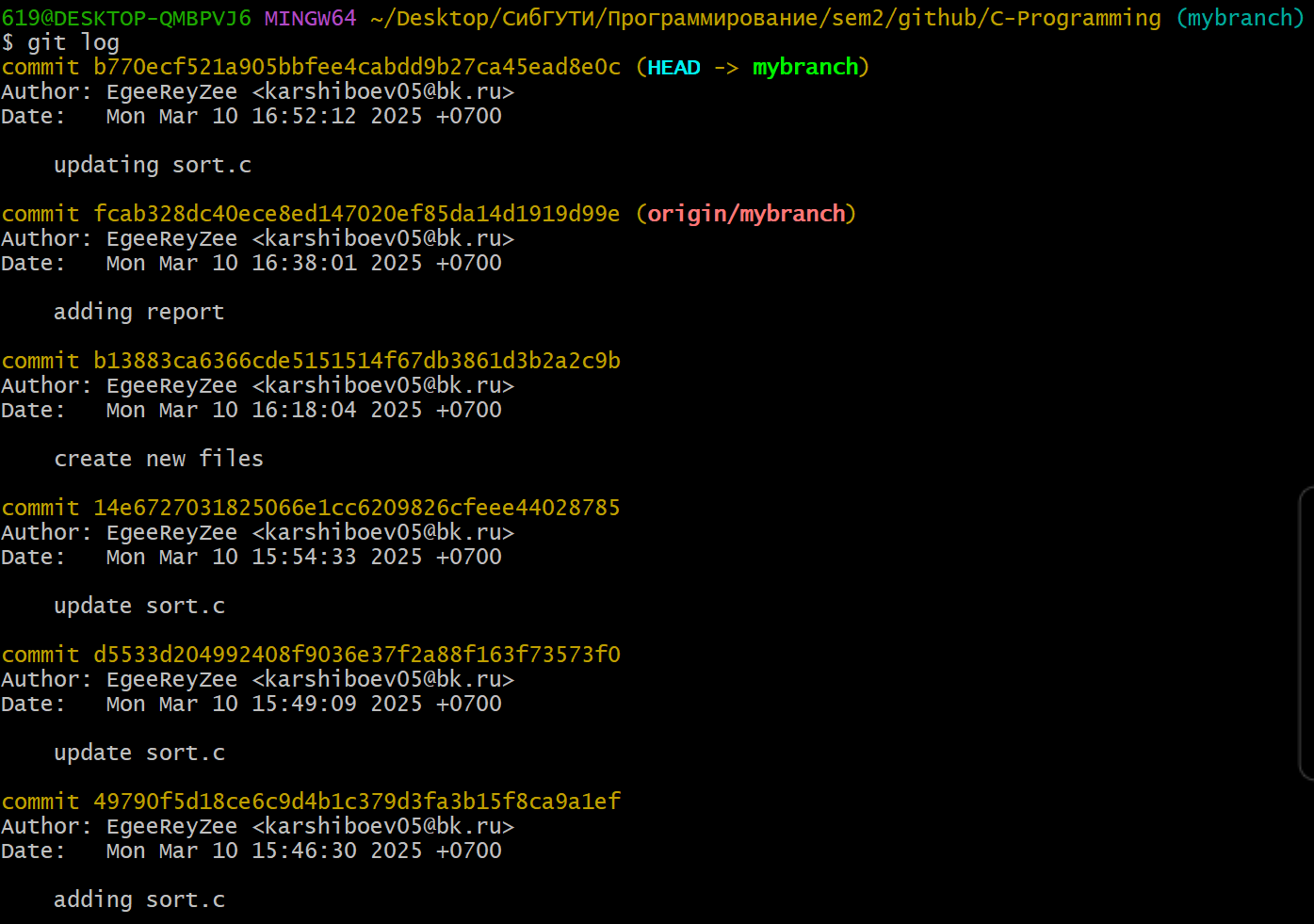


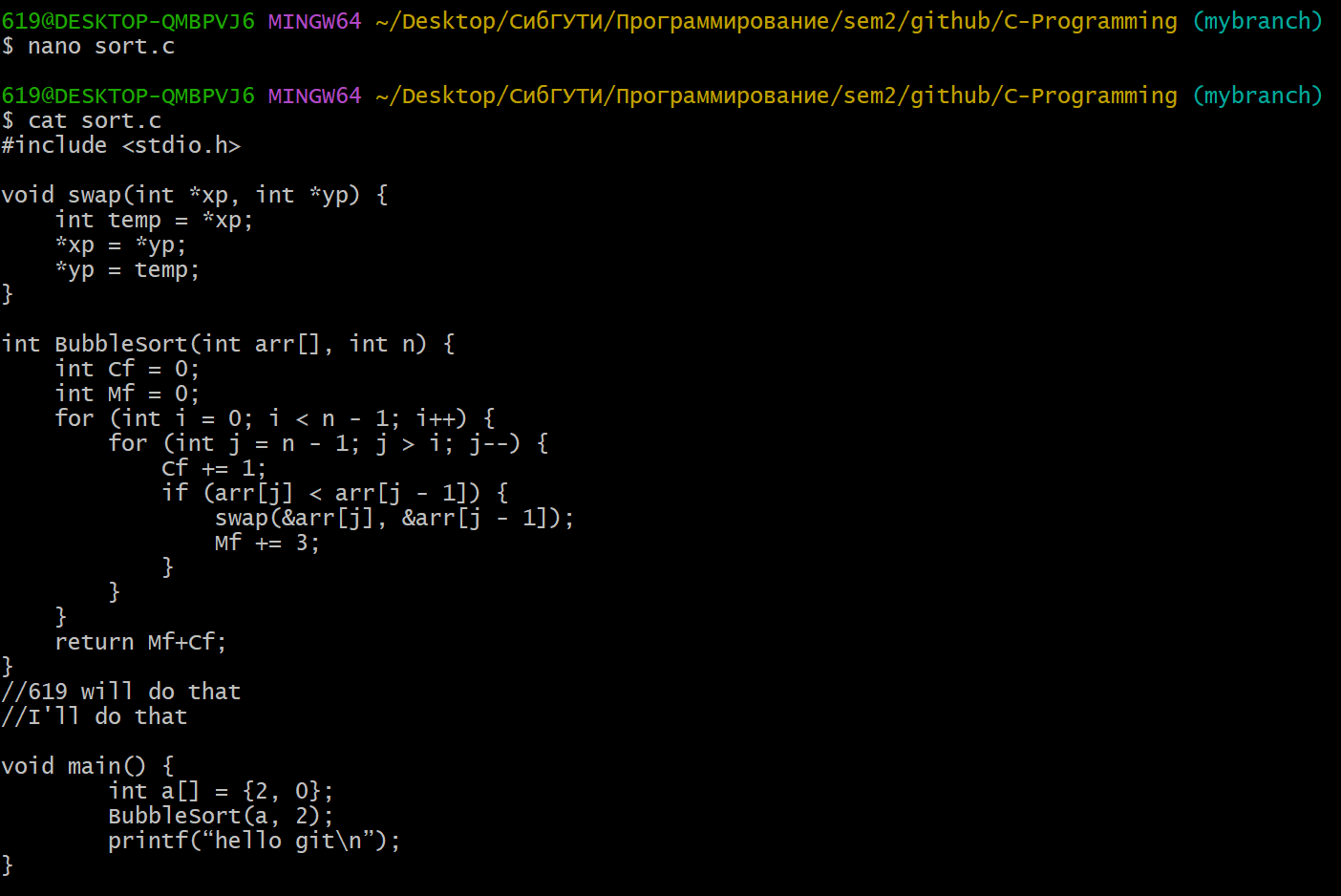
1. Переключитесь на ветку mybranch. В ней будет файл sort.c из предыдущих шагов с функцией сортировки
2. Перезапишите содержимое в sort.c добавив функцию main(), в которой будет объявлен массив из нескольких чисел (пример int a[] = {4, 2, 0};) и вызвана функция сортировки для этого массива.

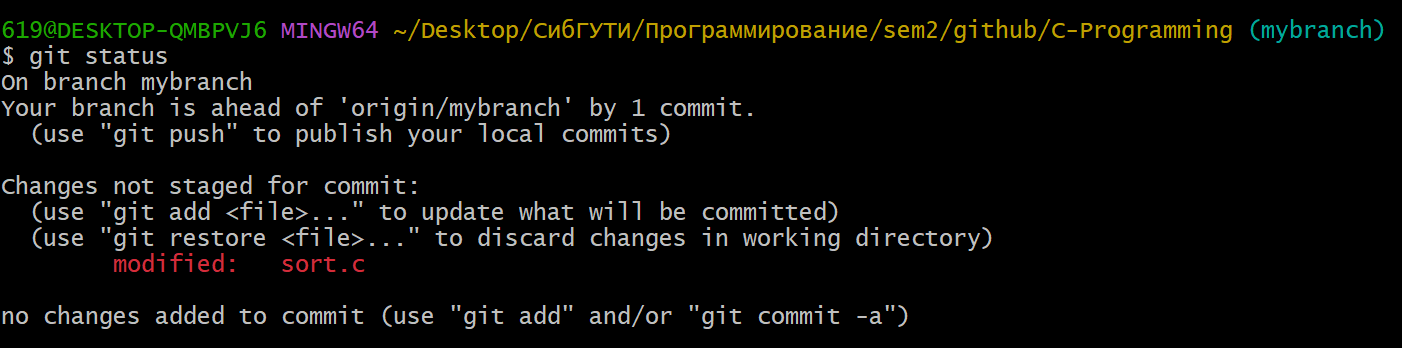
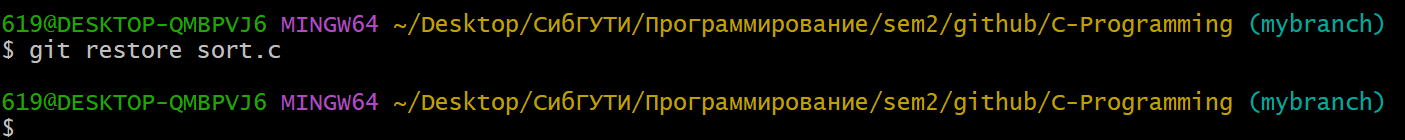
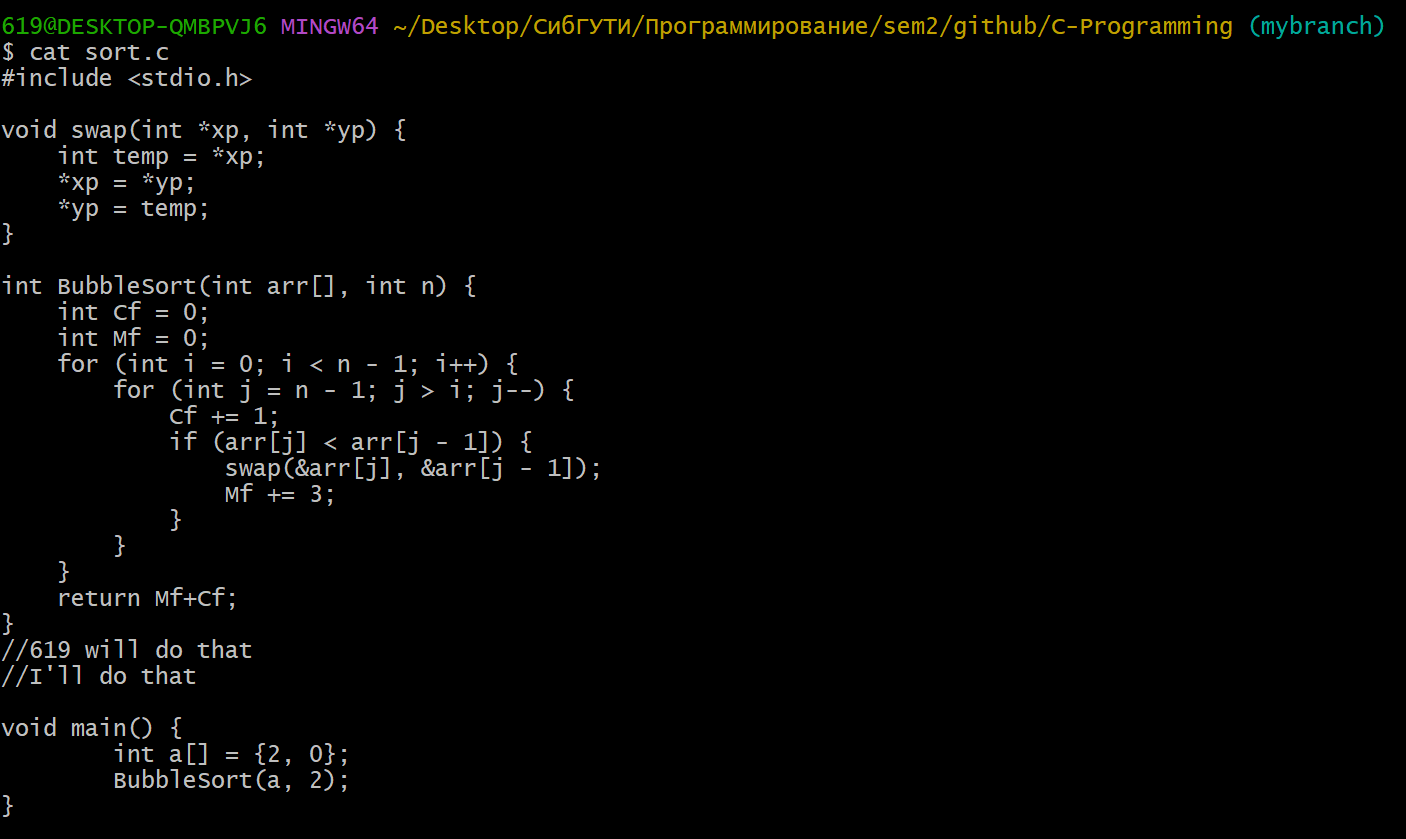
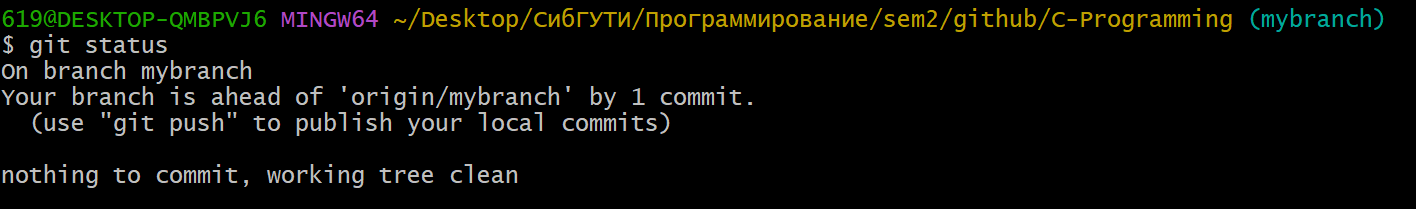
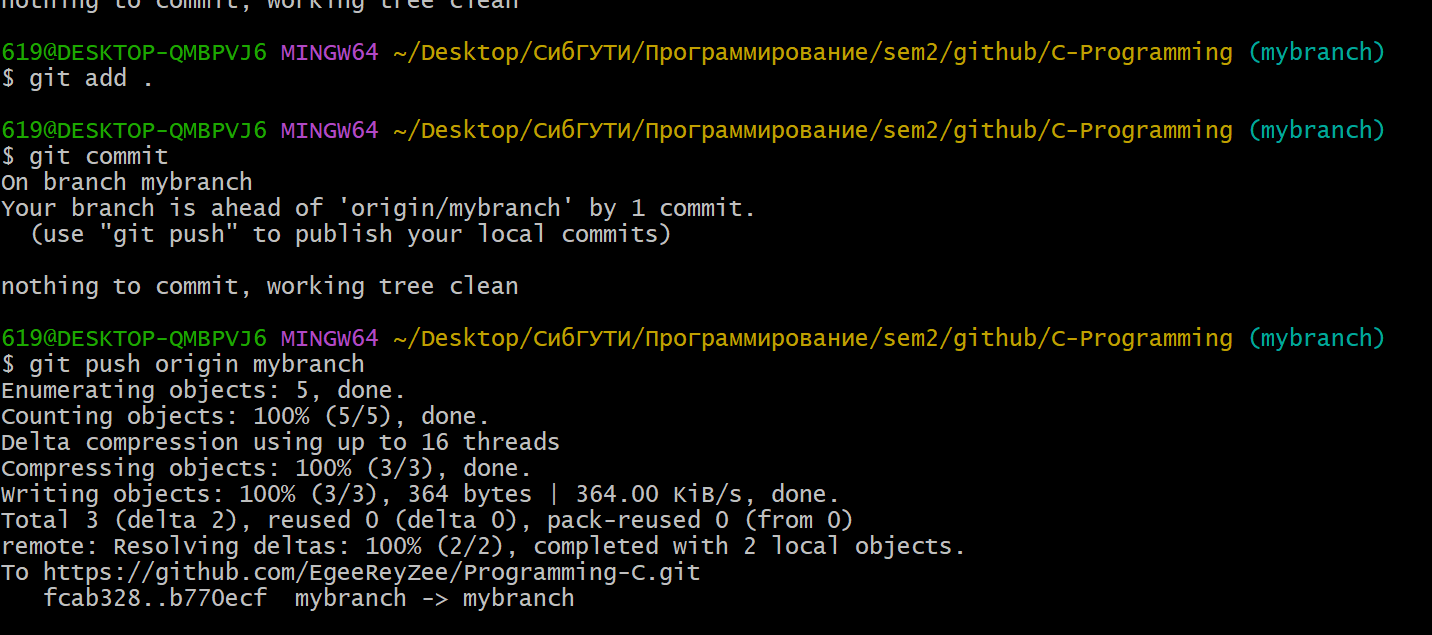


1. Что вам говорит git diff?



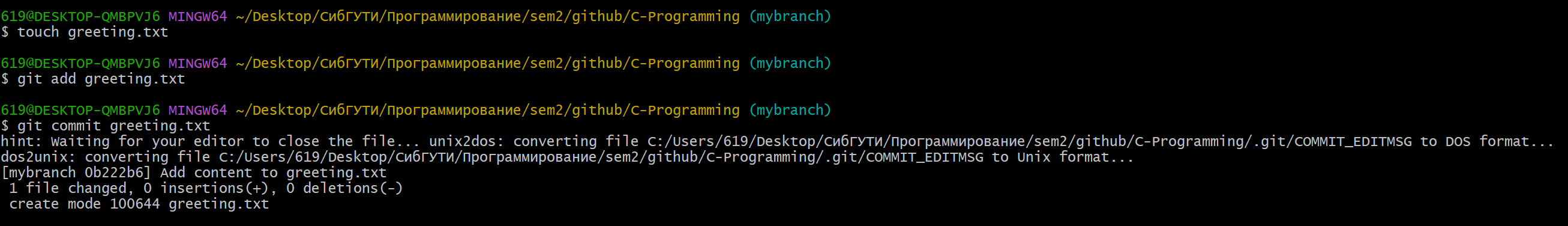
1. Что вам говорит git diff --staged? Пустой? 
2. Добавьте в staged файл sort.c 
3. Что вам говорит git diff?
4. Что вам говорит git diff --staged? 
5. Удалите любое из чисел в массиве в sort.c: 
6. Что вам говорит git diff?
7. Что вам говорит git diff --staged? 
8. Git diff показал какие строки были удалены, а какие добавлены, a git diff –staged показал чем отлич
9. Запустите git status и обратите внимание, что sort.c присутствует дважды в выводе. 
10. Запустите git restore --staged sort.c, чтобы отменить индексацию изменения
11. Что вам теперь говорит git status? 
12. Индексируйте изменение (add) и сделайте коммит 
13. Как выглядит журнал? 
14. Добавьте в sort.c в main() printf(“hello git\n”);.
15. Каково содержимое sort.c?



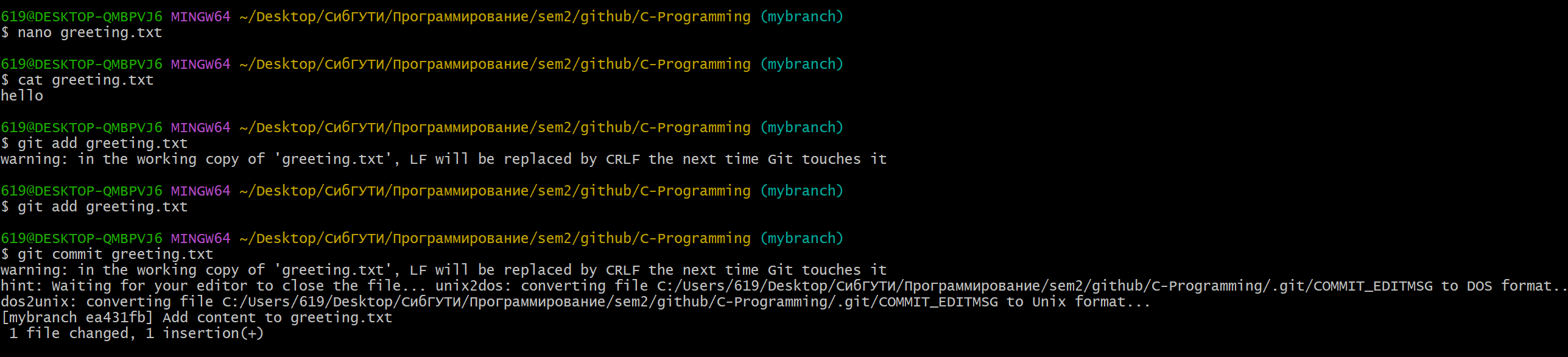
1. Что нам говорит git status? 
2. Запустите git restore sort.c 
3. Каково содержимое sort.c? 
4. Что нам говорит git status? 
5. Запушить на удаленный репо ветку. 

Теперь поиграемся с ветками и ff-merge  
Помните: 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

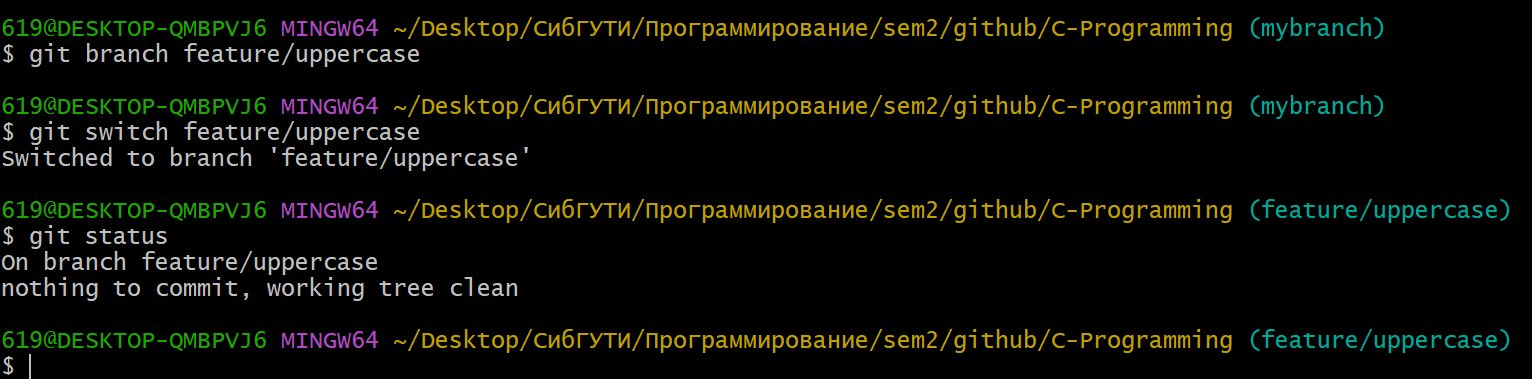
1. Создать файл greeting.txt, проиндексировать его и закоммитить с сообщением “Add file greeting.txt”.



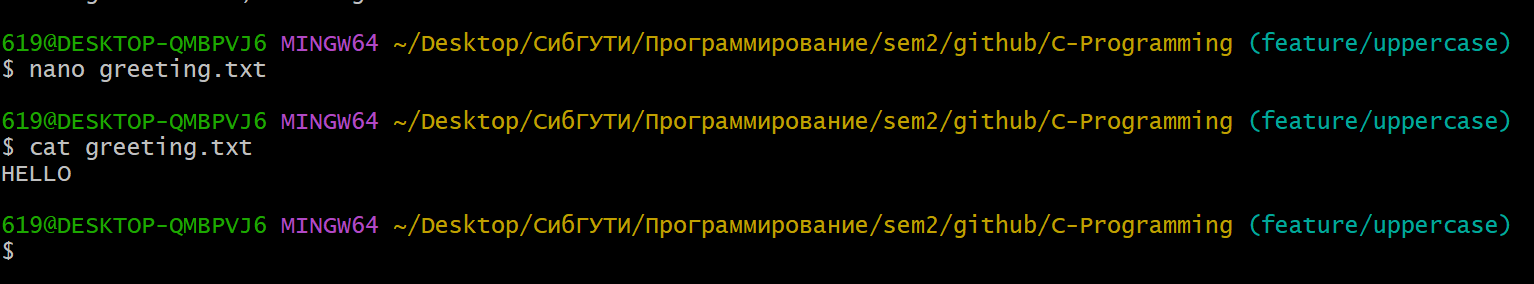
1. Добавить в этот файл слово hello, индексируем и коммитим с текстом "Add content to greeting.txt"



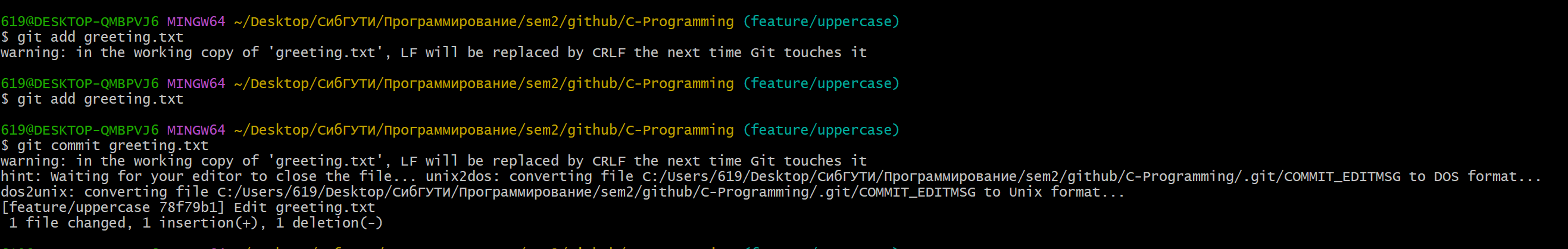
1. Создайте ветку с именем feature/uppercase (да, feature/uppercase — это совершенно допустимое имя ветки и общепринятое соглашение).
2. Переключитесь на эту ветку
3. Каков вывод git status?

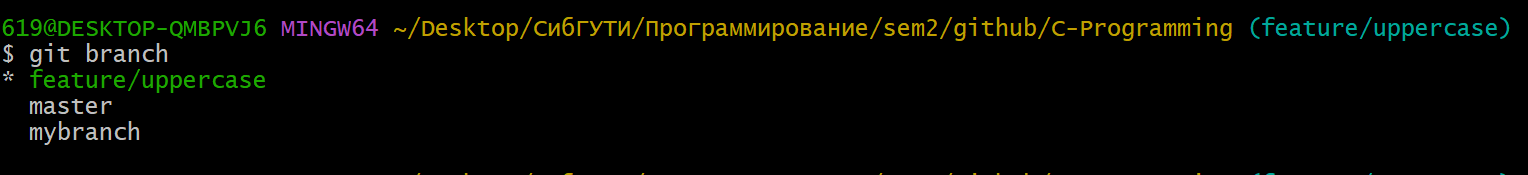
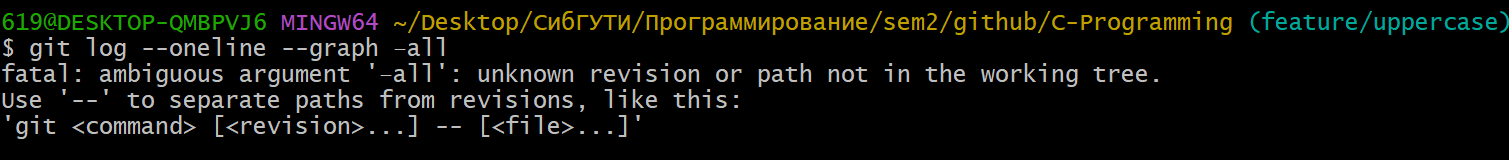
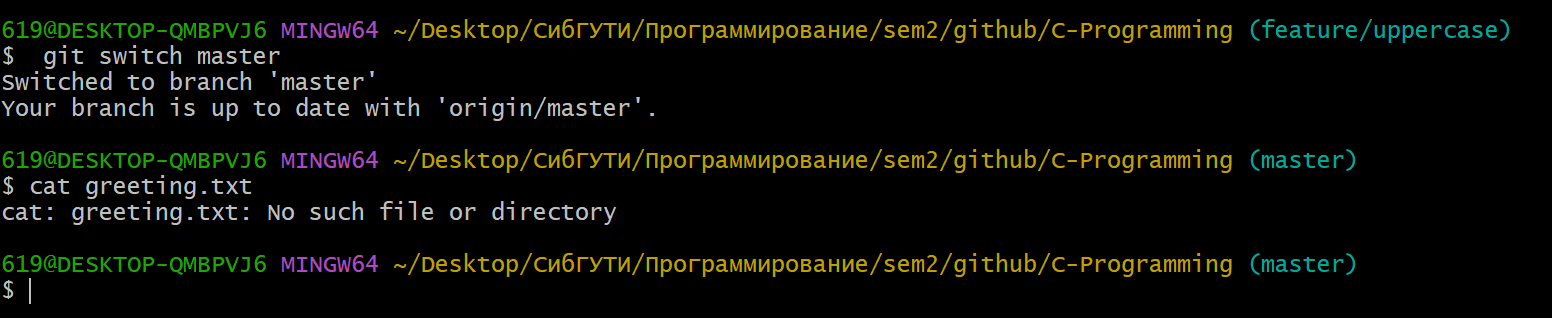


1. Отредактируйте greeting.txt, чтобы он содержал приветствие в верхнем регистре (HELLO)



1. Добавьте файл greeting.txt и закоммитьте



1. Каков вывод git branch? 
2. Каков вывод git log --oneline --graph –all 
3. Переключитесь на главную ветку
4. Используйте cat, чтобы увидеть содержимое файла greetings.txt 
5. Сравните ветки
6. Объедините ветки 