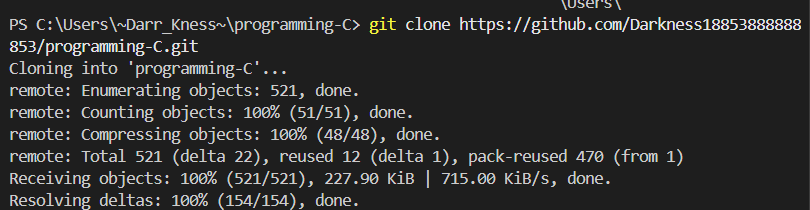
Практическое задание 3. Git

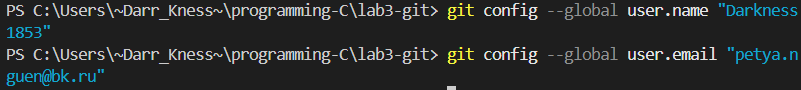
1)Клонируем репозиторий к себе.



2)Создаем в репозитории папку для лабы



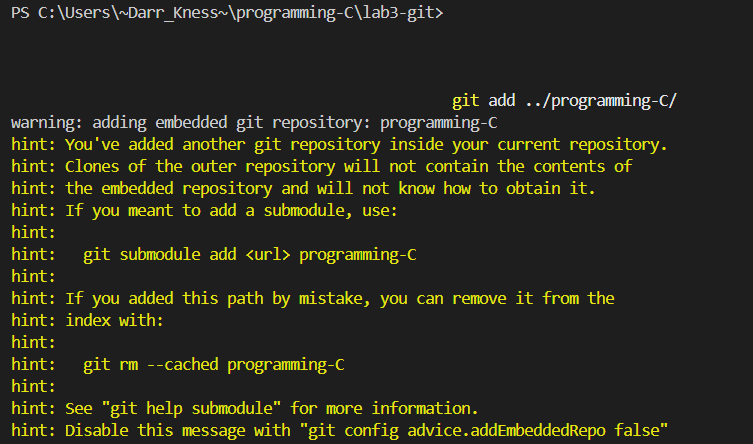
3)Логинимся

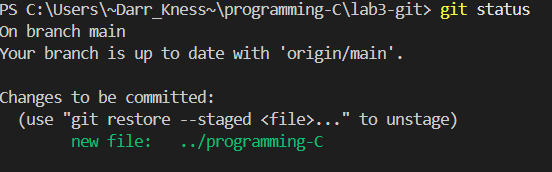
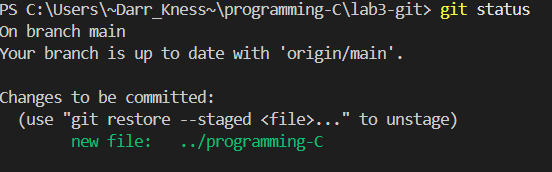


4)Выбираем текстовый редактор

На оценку 3:

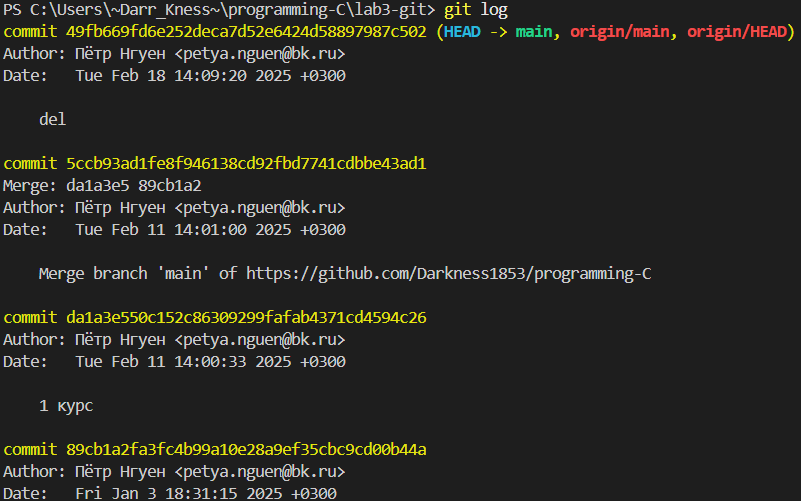
Добавляем папку в индекс(staging area) , чтобы Git начал отслеживать изменения в ней.

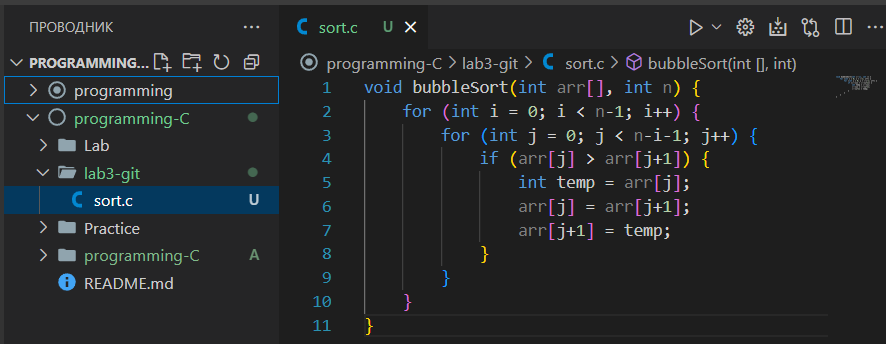
1)Используемgit status, чтобы узнать, на какой ветке мы находимся.

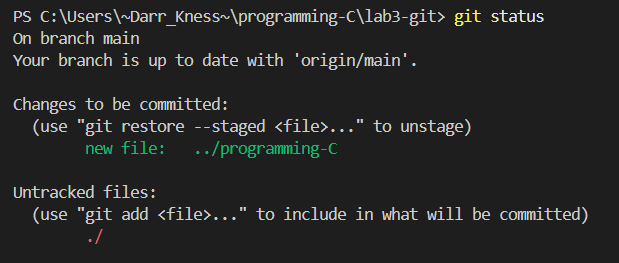
2)С помощью git log просмотрели истории коммитов в репозитории. Эта команда показывает список всех коммитов, начиная с последнего (самого свежего) и заканчивая самым первым коммитом в репозитории.

**Хэш коммита** (commit hash) — уникальный идентификатор коммита

1. **Автор** (Author) — имя и email автора коммита.
2. **Дата** (Date) — дата и время создания коммита.
3. **Сообщение коммита** (commit message) — текст, который был указан при создании коммита

3)Создаем файл sort.c и вставляем код функции сортировки.

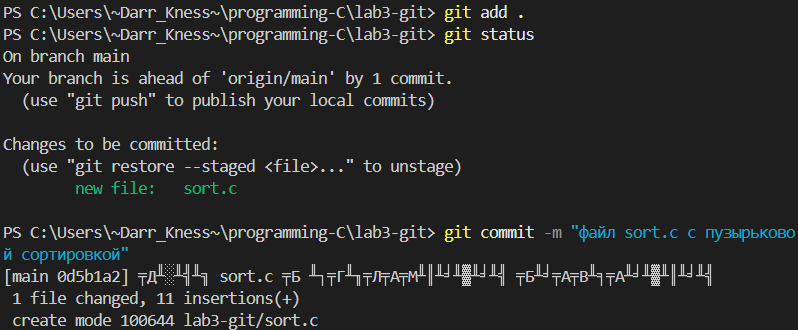
4)Вывод git status показало, что моей рабочей директории есть неотслеживаемые файлы. Эти файлы ещё не добавлены в staging area. Git предлагает использовать команду git add <file>, чтобы начать отслеживать их и включить в следующий коммит.

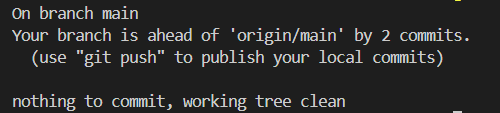
5)Добавим файл в область stage (add)

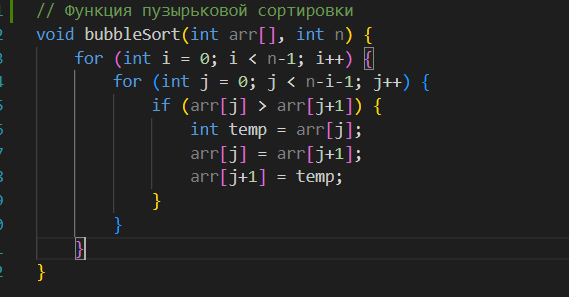
Добавим неотслеживаемые файлы в staging area:git add .(. — добавим все неизвестные файлы)

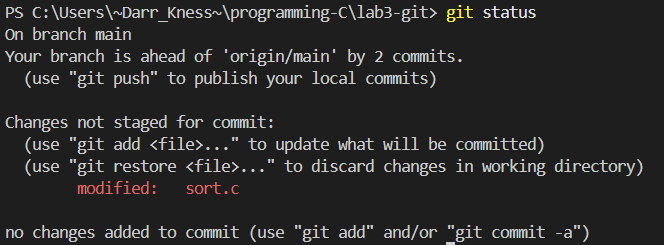
Проверьте статус снова: git status

Создайте коммит: git commit -m "файл sort.c с пузырьковой сортировкой"

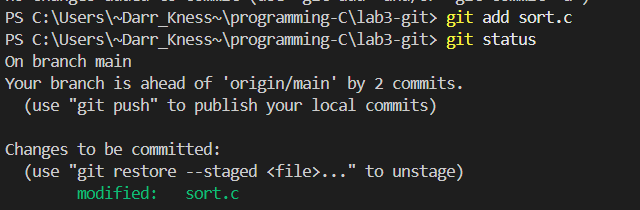
6)Проверим статус:

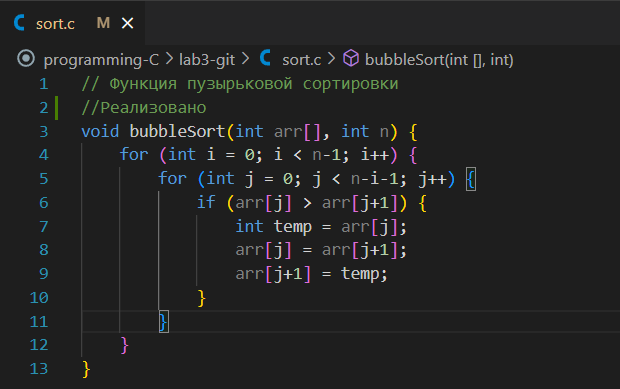
7)Добавили комментарий в созданный файл

10)Вот как выглядит git status

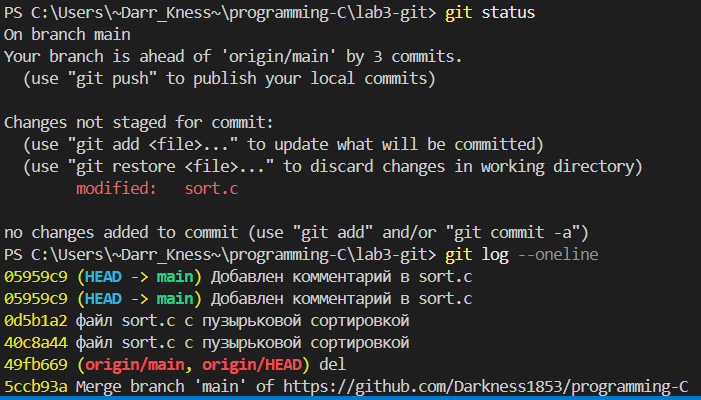
11)Добавим (add) изменение файла и

12)вызовем git status

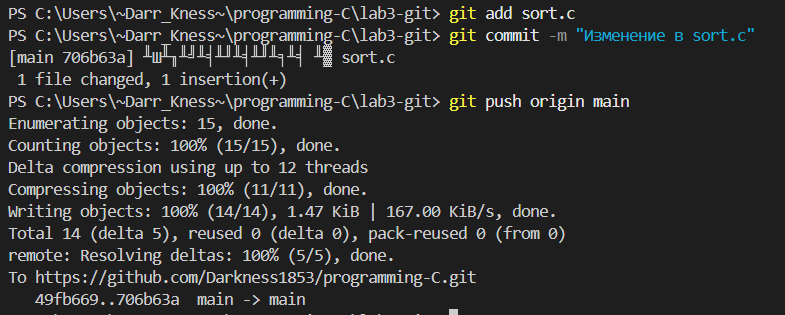
13)Изменим ещё раз файл

14)Проверим статус git status

15)И журнал с помощью команд git log --oneline

16) Добавим в stage и закоммитим последнее изменение

17) Запушим на удаленный репо (git push)

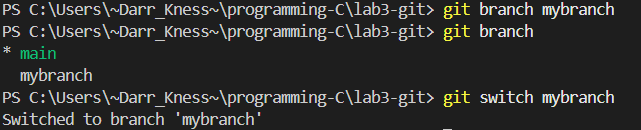


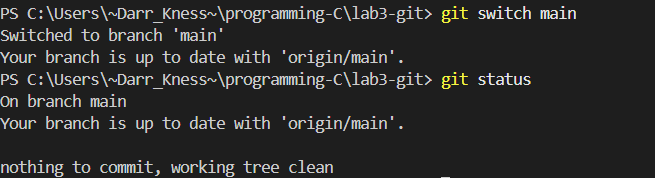
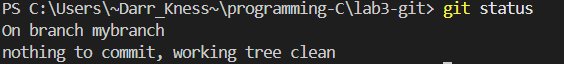
Работа с ветками

1)Создадим новую ветку с помощью команды git branch mybranch

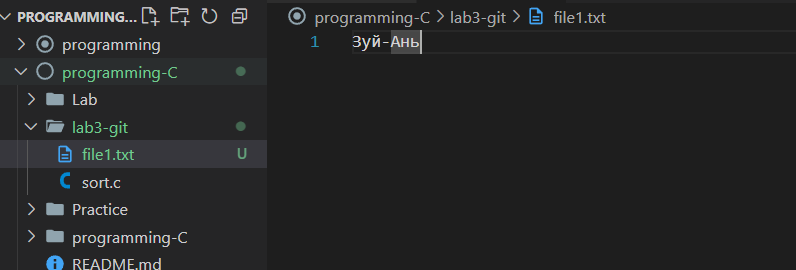
2)Проверим список git branch

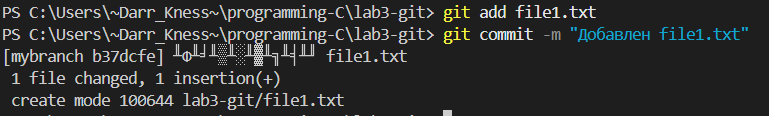
3)Переключимся на новую ветку

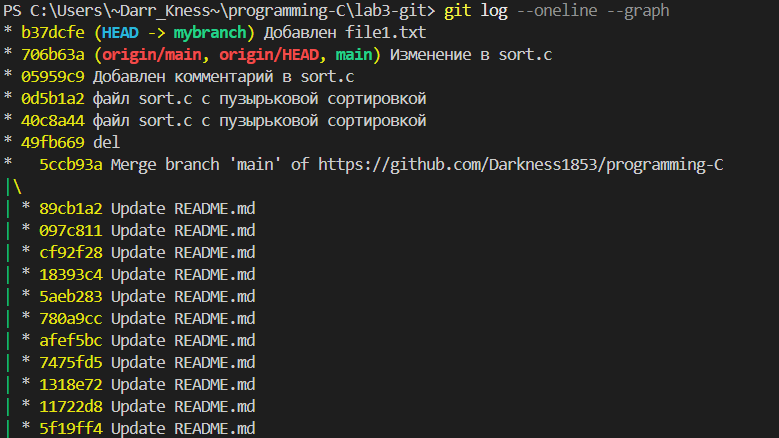
4)Git работает с ветками. В ветке mybranch нет изменений, и рабочая директория чиста, также как и в ветке main (нет изменённых или неотслеживаемых файлов).

5)Вернулись в ветку mybranch

6)Создадим файл file1.txt и добавим в него свое имя.

7)Добавим файл и закоммитим это изменение.

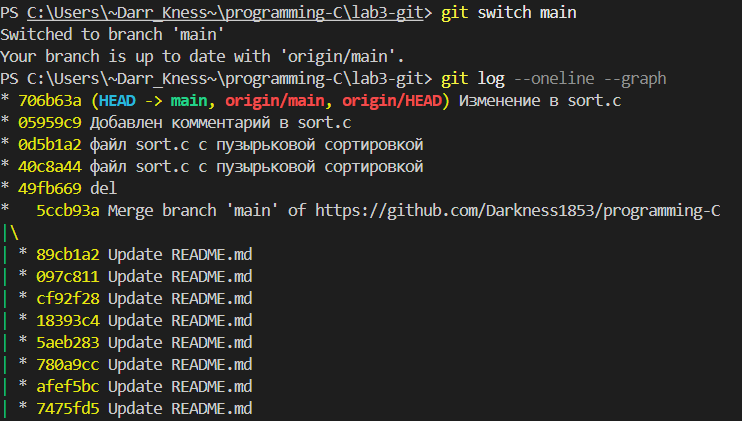
8)Просмотрим журнал

9)Вернемся к ветке с именем master.

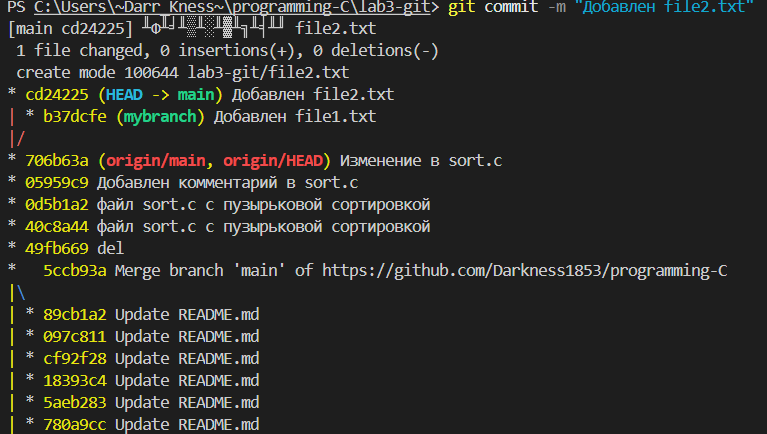
10)Используем git log --oneline —graph

### Что изменилось:

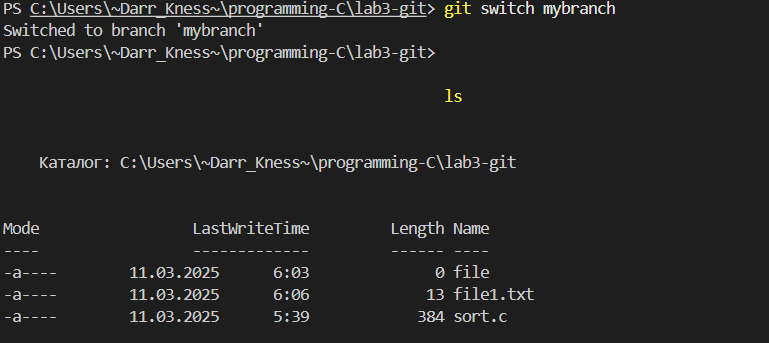
1. **Положение**HEAD:
   * В первом выводе HEAD указывал на ветку mybranch, а во втором — на main.
2. **Последний коммит**:
   * В mybranch последний коммит — b37dcfe (добавление file1.txt).
   * В main последний коммит — 706b63a (изменение в sort.c).
3. **Наличие коммитов**:
   * Коммит b37dcfe виден только в ветке mybranch, так как он был сделан в этой ветке и ещё не был слит с main.
4. **Граф истории**:
   * Граф истории остался практически одинаковым, за исключением того, что в mybranch добавлен один дополнительный коммит (b37dcfe).

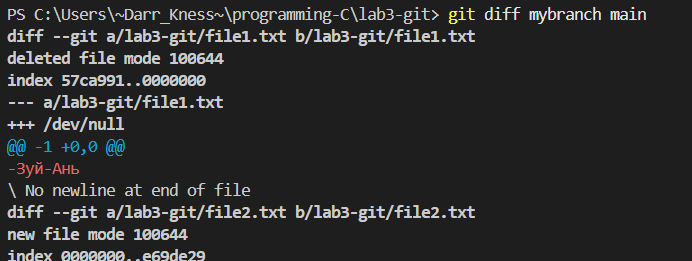
11)Создадим новый файл с именем file2.txt и закоммитим его.

12)Используя git log --oneline --graph –all, увидели, что моя ветка указывает на новый коммит, и что теперь у двух веток разные коммиты.

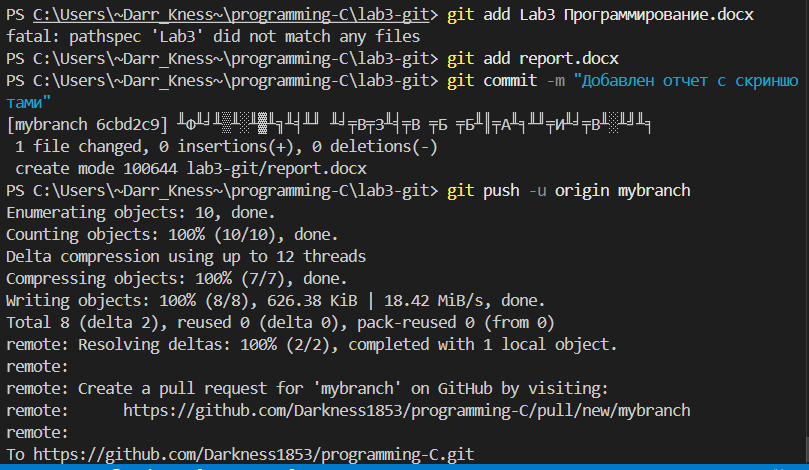
13)Переключимся на мою ветку mybranch.

14)Наш файл file2.txt пропал

15)Используя git diff mybranch main Сравним обе ветки

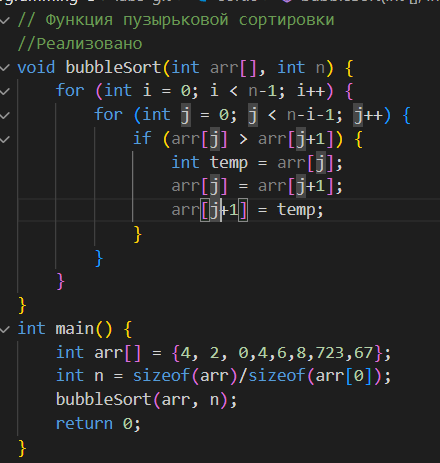
16)Добавим текстовый документ со скриншотами в ветку **mybranch**. Закоммитим и запушим на удаленный репозиторий ветку mybranch (git push -u origin mybranch)

Убедимься что в github.com две ветки master и mybranch. Не забудем запушить изменения master ветки в main

На оценку 4

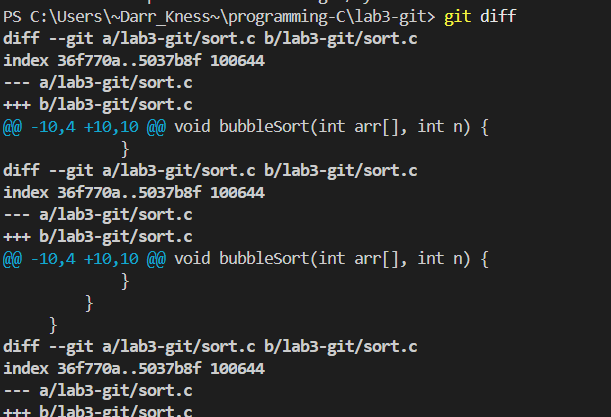
1)Переключимся в ветку mybranch

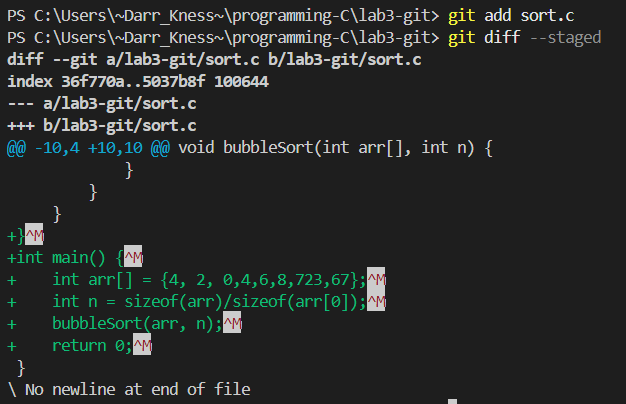
2)Перезапишем содержимое в sort.c добавив функцию main(), в которой будет объявлен массив из нескольких чисел (пример int a[] = {4, 2, 0};) и вызовем функция сортировки для этого массива.



3)Вывод команды git diff показывает изменения, которые были сделаны в файле sort.c, но ещё не были добавлены в staging area (индекс) с помощью git add.

Команда git diff сравнивает текущее состояние рабочей директории с последним закоммиченным состоянием (или состоянием в staging area, если файл был добавлен туда).в файл были добавлены новые строки. Они обозначены символом + (новые строки) и ^M (символ возврата каретки)



4)Команда git diff --staged показывает изменения, которые были добавлены в **staging area** (индекс) с помощью git add, но ещё не закоммичены.

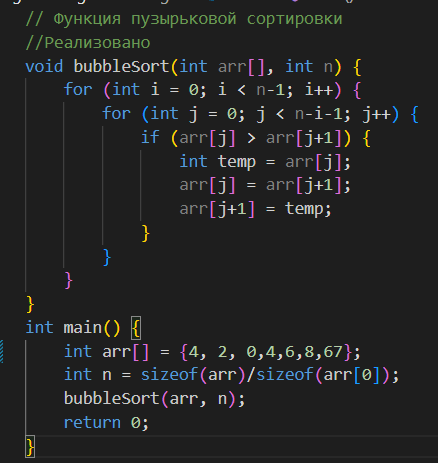
вывод команды **не пустой**, и он показывает, что файл lab3-git/sort.c был изменён и добавлен в staging area.

5)Добавили в staged файл sort.c

6) git diff выводит, что нет **изменений** в файле

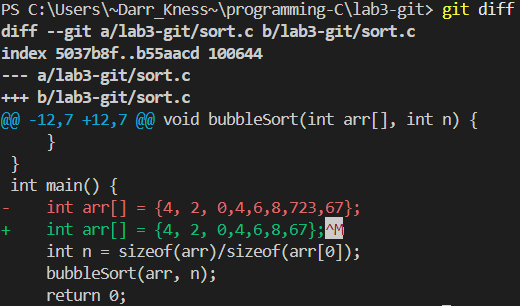
7) git diff —staged выводит, что в staging area находится изменённая версия файла sort.c

8)Удалим одно число из массива в sort.c



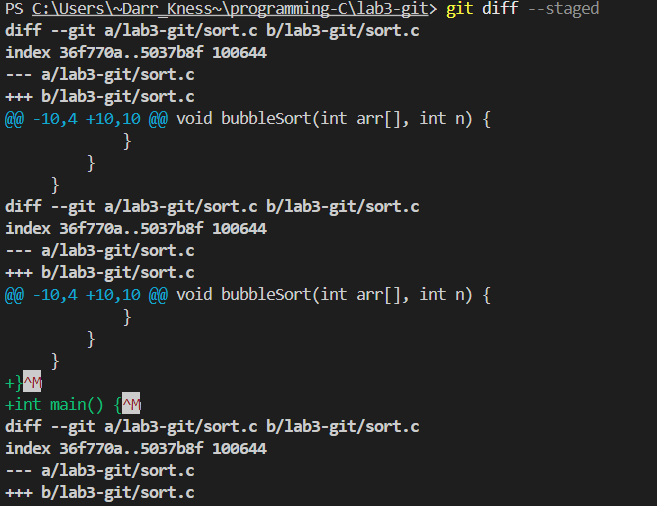
9)Проверим изменения с помощью команды git diff

Видим, что в файле была изменена строка, содержащая массив arr в функции main()



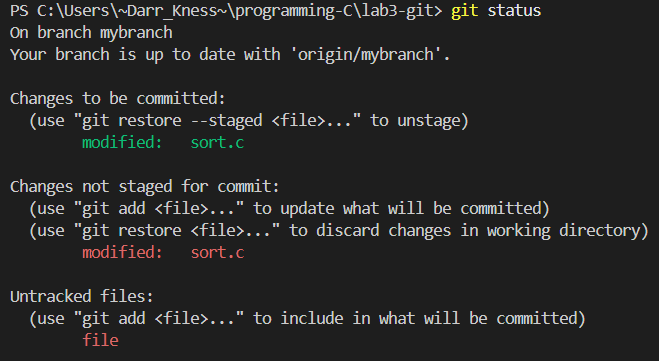
10)Выполним git diff —staged

В файл была добавлена новая функция main()

11)Обьясним, что происходит:Команда git diff --staged показывает изменения, которые были добавлены в **staging area** (индекс) с помощью git add, но ещё не закоммичены.

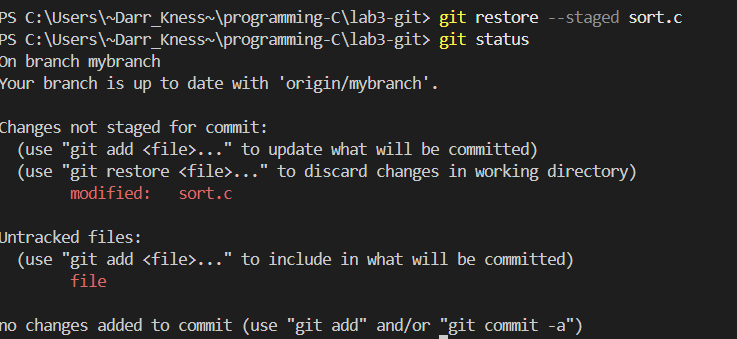
В файл была добавлена новая функция main()

12)Запустили git status и обратили внимание, что sort.c присутствует дважды в выводе.

13)Запустим git restore --staged sort.c , чтобы отменить индексацию изменения

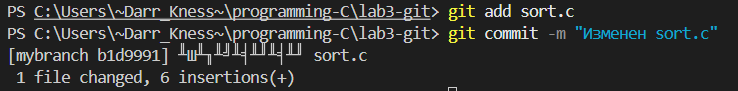
Команда git restore --staged убирает файл из staging area (индекса), но **сохраняет изменения в рабочей директории**.

* После выполнения команды файл sort.c больше не готов к коммиту, но изменения в нём остаются.

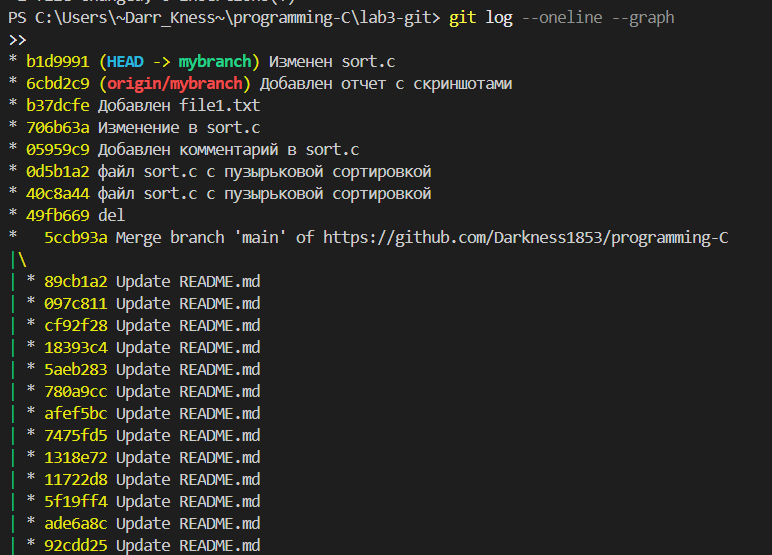
14)Файл sort.c всё ещё изменён в рабочей директории, но эти изменения не добавлены в staging area.

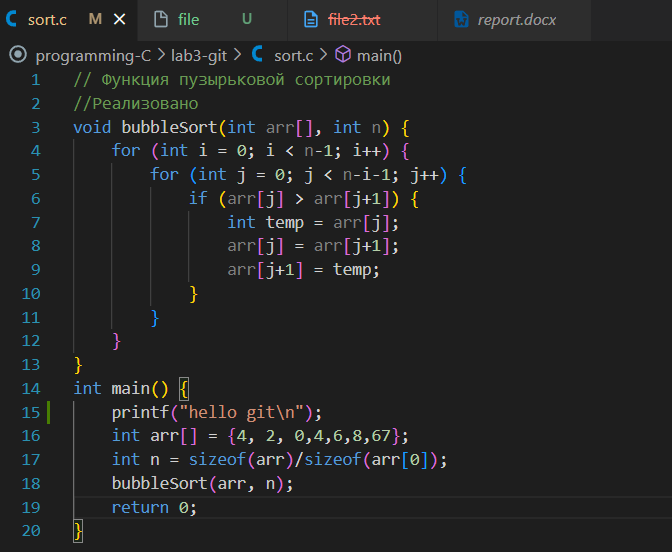
* **Untracked files**:
  + Файл file по-прежнему не отслеживается Git.
* **no changes added to commit**:
  + Нет изменений, готовых к коммиту (staging area пуст)

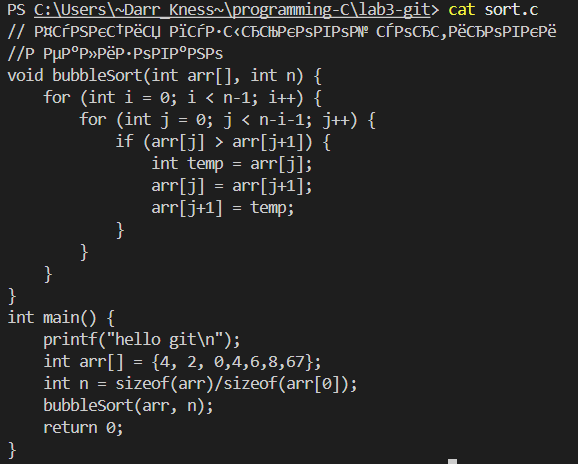
15)Индексируем изменения (add) и делаем коммит

16) Журнал показывает нам, что был последнее изменение было в файле sort.c

и что мы находимся в mybranch.

17)Добавим в sort.c в main() printf(“hello git\n”)

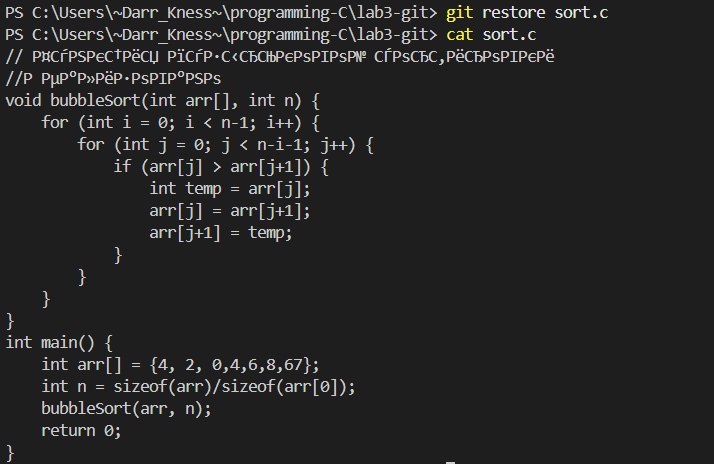
18)Содержимое sort.c



20)Запустим git restore sort.c

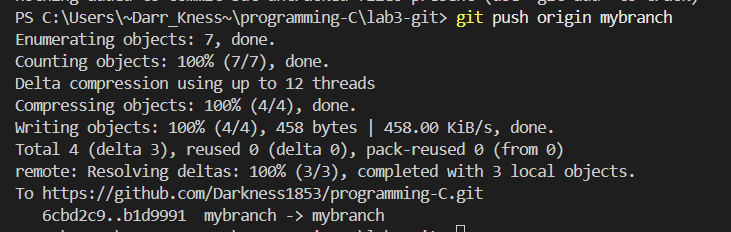
git restore sort.c вернула файл к состоянию последнего коммита, и изменения, которые не были закоммичены, были отменены.

21)Содержимое файла sort.c

22)git status говорит нам, что:

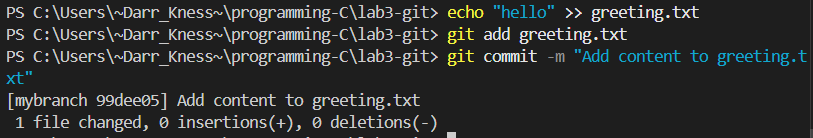
* Мы находитмься на ветке mybranch.
* Моя локальная ветка опережает удалённую ветку origin/mybranch на **1 коммит**.
* В моей рабочей директории есть неотслеживаемый файл file.
* Нет изменений, готовых к коммиту (staging area пуст).

23)Запушитм на удаленный репо ветку с помощью команды git push origin mybranch

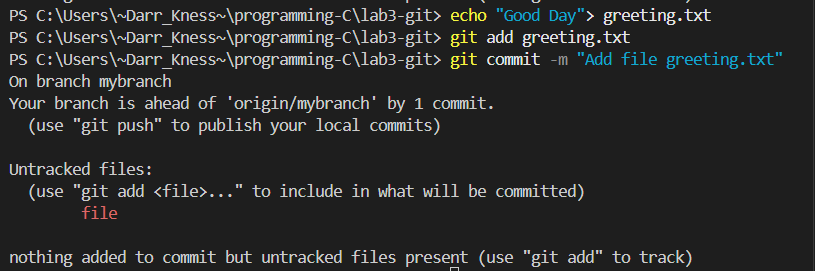


Ветки:

1)Создадим файл greeting.txt и закоммитим

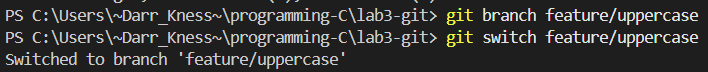


2)Добавим в этот файл слово hello, индексируем и коммитим с текстом "Add content to greeting.txt"

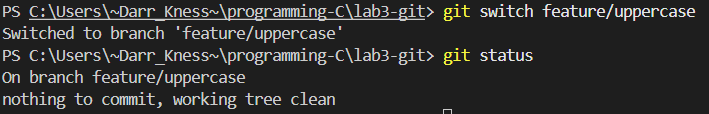


3)Создадим ветку с именем feature/uppercase с помощью команды git branch feature/uppercase

4)Переключимся в нее

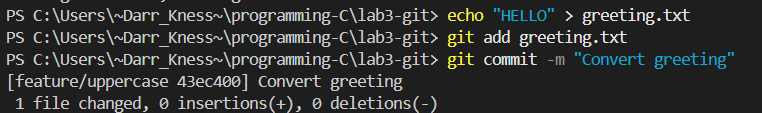


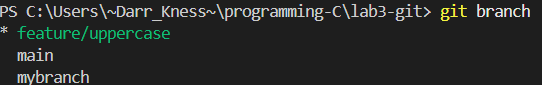
5)Вывод git status



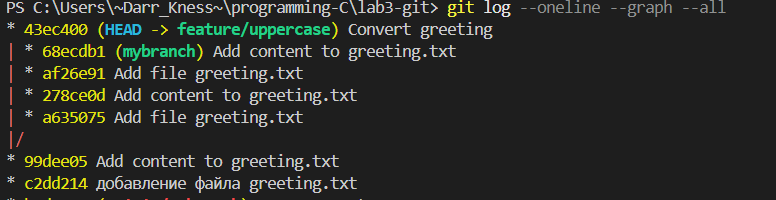
6)Отредактируем greeting.txt, чтобы он содержал приветствие в верхнем регистре (HELLO)

7)Добавим файл greeting.txt и закоммитим

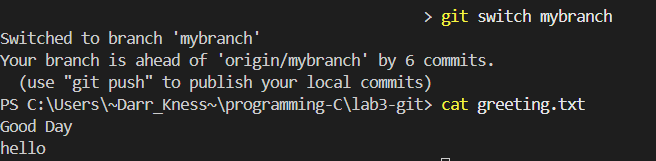
8) git branch выводит информацию:pвёздочка (\*) указывает на текущую ветку(main).

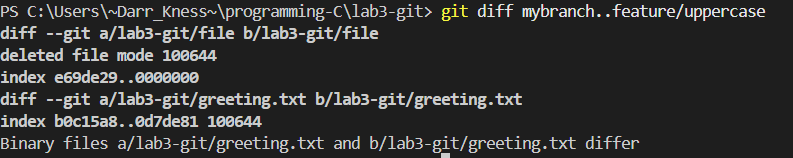
Ветка feature/uppercase существует.

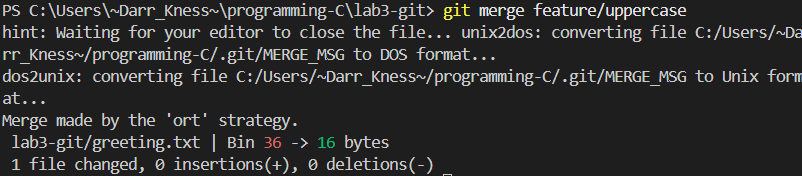
9)Выводим git log --oneline --graph –all

10)Переключимся на главную ветку git switch mybranch

11)Выведем содержимое файла cat greeting.txt

12)Сравним ветки git diff mybranch..feature/uppercase

13)Объединим ветки

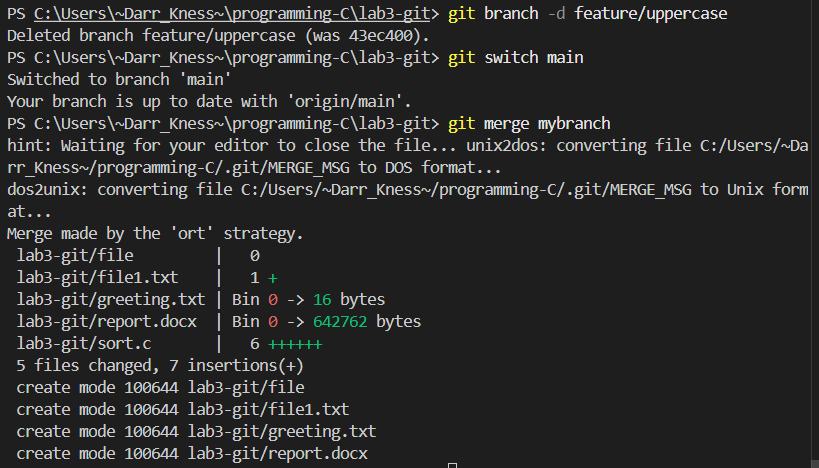
14)Используем cat, чтобы увидеть содержимое файла greetings.txt



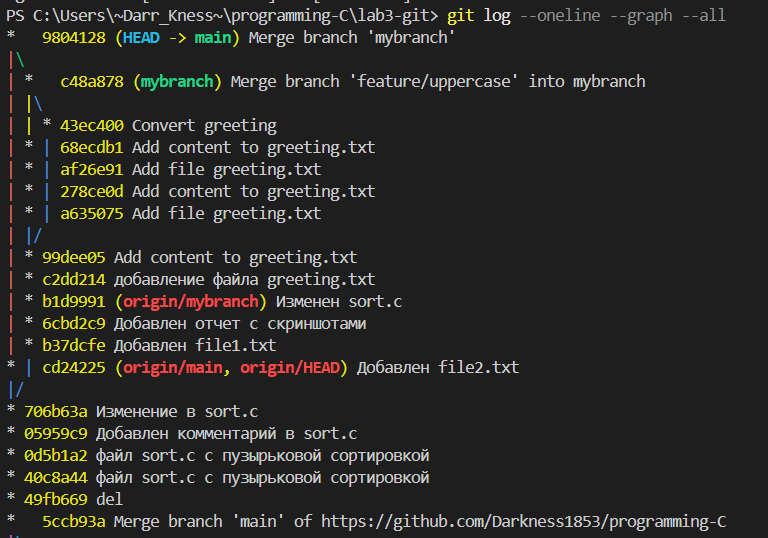
15)Удалим ветку с заглавными буквами (feature/uppercase) с помощью команды

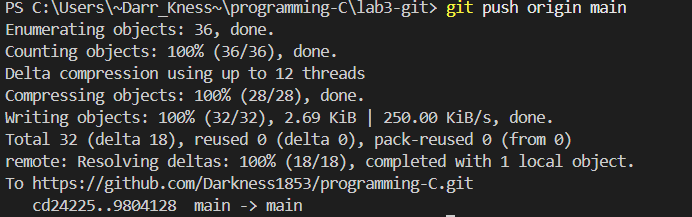
git branch -d feature/uppercase

16)Смержим ветку mybranch в master (git merge)



17) git log --oneline --graph --all выводит

18)Запушим изменения ветки master на удаленный репо. git push origin main



19)Запушим документ с результатами вашей работы