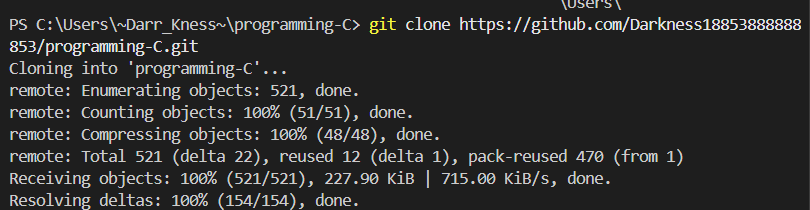
Практическое задание 3. Git

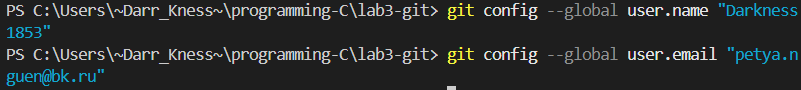
1)Клонируем репозиторий к себе.



2)Создаем в репозитории папку для лабы



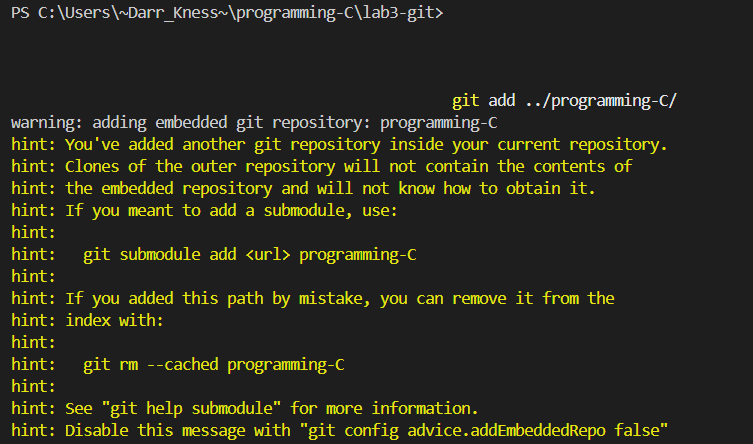
3)Логинимся

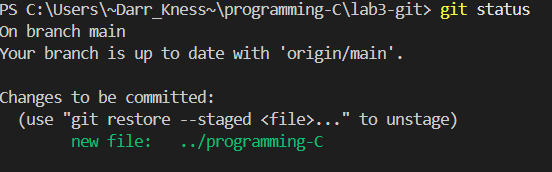
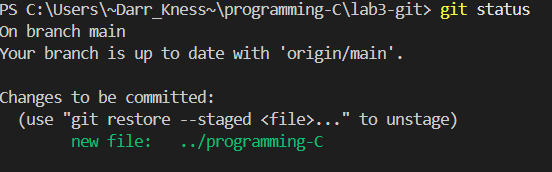


4)Выбираем текстовый редактор

На оценку 3:

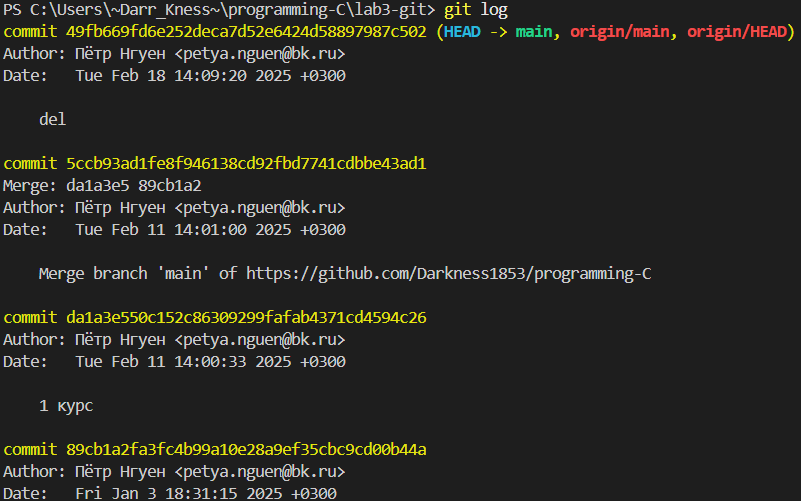
Добавляем папку в индекс(staging area) , чтобы Git начал отслеживать изменения в ней.

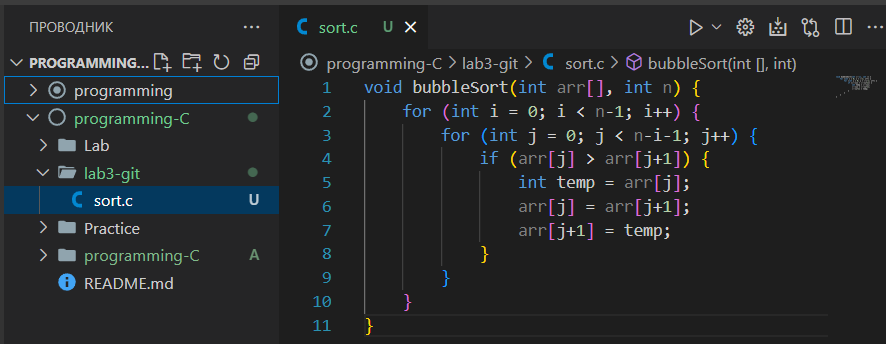
1)Используемgit status, чтобы узнать, на какой ветке мы находимся.

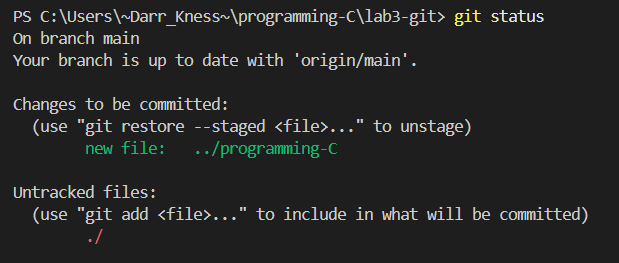
2)С помощью git log просмотрели истории коммитов в репозитории. Эта команда показывает список всех коммитов, начиная с последнего (самого свежего) и заканчивая самым первым коммитом в репозитории.

**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.

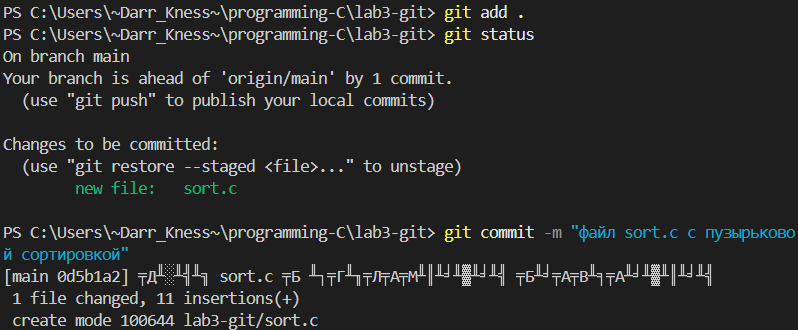
4)Вывод git status показало, что моей рабочей директории есть неотслеживаемые файлы. Эти файлы ещё не добавлены в staging area. Git предлагает использовать команду git add <file>, чтобы начать отслеживать их и включить в следующий коммит.

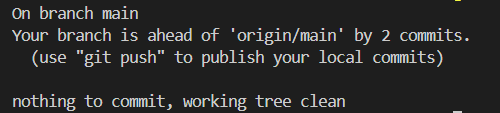
5)Добавим файл в область stage (add)

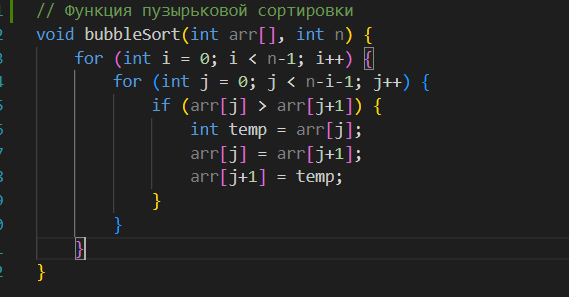
Добавим неотслеживаемые файлы в staging area:git add .(. — добавим все неизвестные файлы)

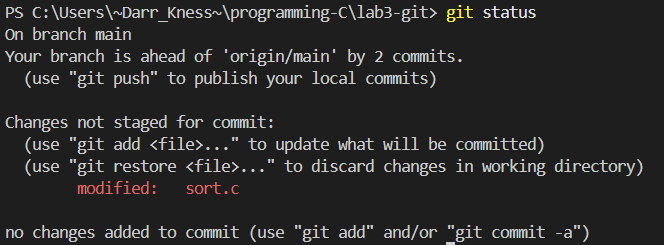
Проверьте статус снова: git status

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

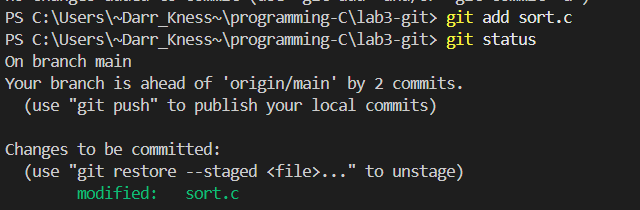
6)Проверим статус:

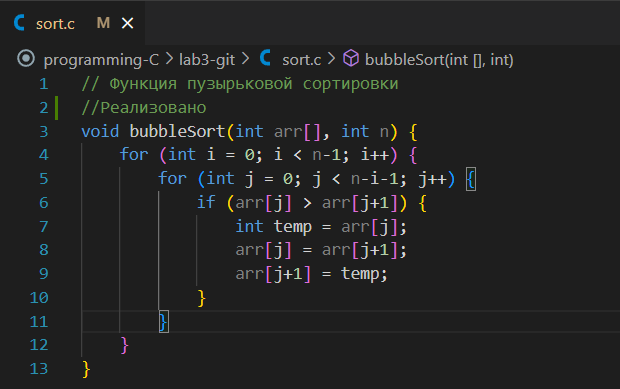
7)Добавили комментарий в созданный файл

10)Вот как выглядит git status

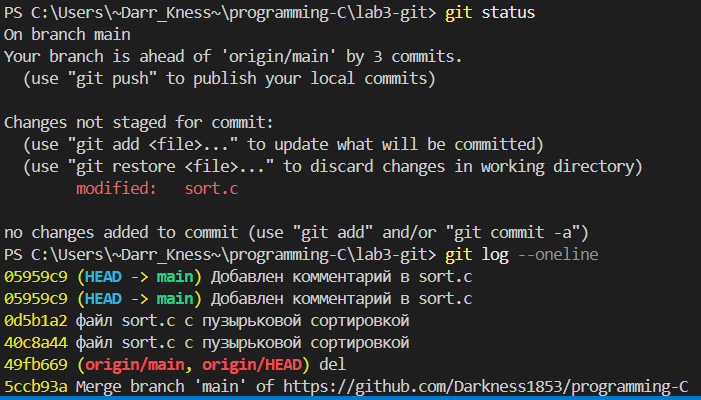
11)Добавим (add) изменение файла и

12)вызовем git status

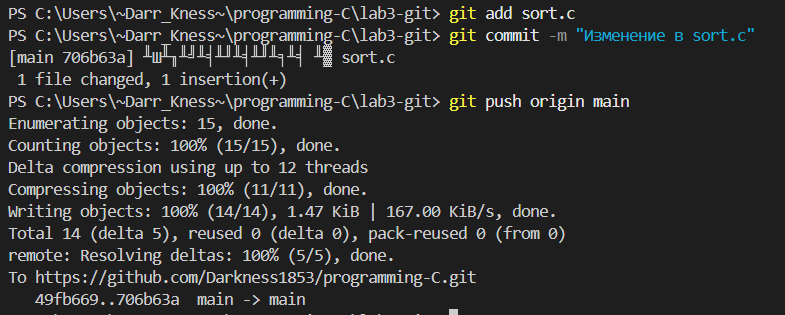
13)Изменим ещё раз файл

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)

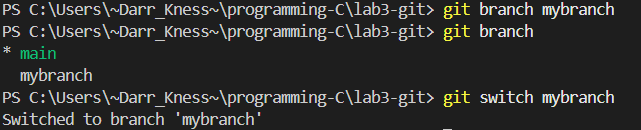


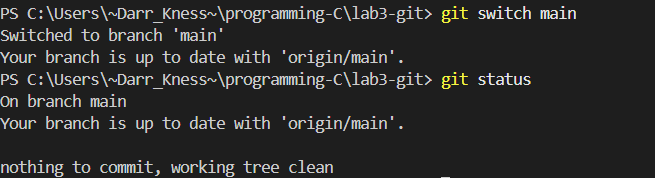
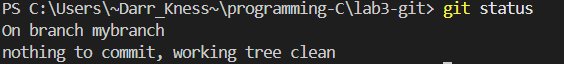
Работа с ветками

1)Создадим новую ветку с помощью команды git branch mybranch

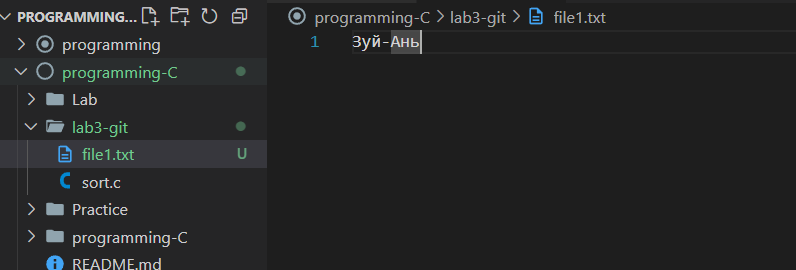
2)Проверим список git branch

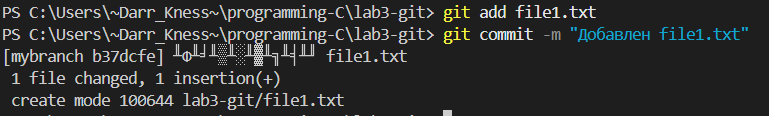
3)Переключимся на новую ветку

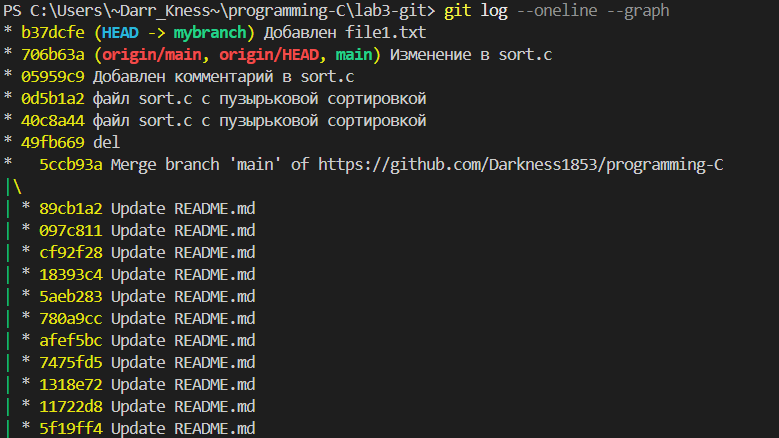
4)Git работает с ветками. В ветке mybranch нет изменений, и рабочая директория чиста, также как и в ветке main (нет изменённых или неотслеживаемых файлов).

5)Вернулись в ветку mybranch

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

7)Добавим файл и закоммитим это изменение.

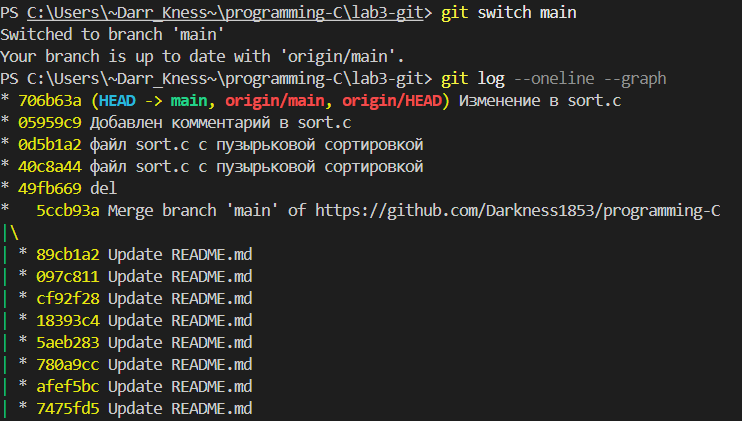
8)Просмотрим журнал

9)Вернемся к ветке с именем master.

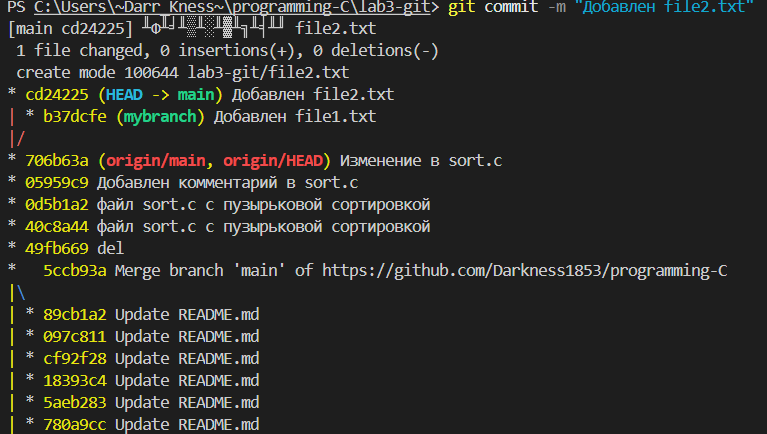
10)Используем git log --oneline —graph

### Что изменилось:

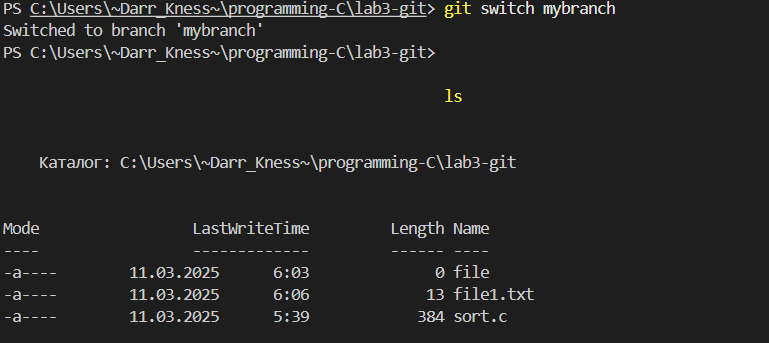
1. **Положение**HEAD:
   * В первом выводе HEAD указывал на ветку mybranch, а во втором — на main.
2. **Последний коммит**:
   * В mybranch последний коммит — b37dcfe (добавление file1.txt).
   * В main последний коммит — 706b63a (изменение в sort.c).
3. **Наличие коммитов**:
   * Коммит b37dcfe виден 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).

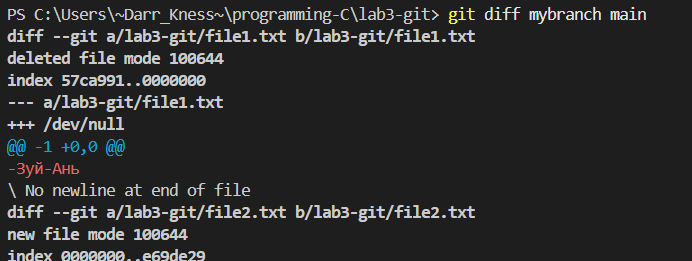
11)Создадим новый файл с именем file2.txt и закоммитим его.

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