Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования «Ульяновский государственный технический университет»

Кафедра «Вычислительная техника»

**Отчет по лабораторной работе №5**

Дисциплина: «Разработка профессиональных приложений»

GIT

Выполнил:

студент группы ИВТАСбд-21

Абросимов К. С.

Проверил:

преподаватель кафедры

«Вычислительная техника»

Исхаков И.И.

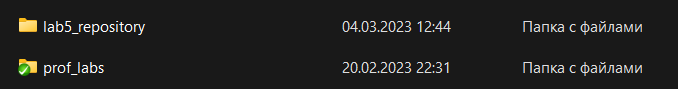
Ульяновск, 2023

1. **Задание по варианту**

Выполнить данные задания по GIT с помощью консоли.

* 2.1. Создать локальный репозиторий
* 2.2. Добавить несколько файлов
* 2.3. Сделать один из файлов игнорируемым
* 2.4. Зафиксировать изменения
* 2.5. Внести изменения в файлы и зафиксировать изменения
* 2.6. Создать несколько бранчей, редактировать файлы в разных бранчах и фиксировать изменения. Осуществить слияние двух веток в одну.
* 2.7. Создать аккаунт github (если его нет, а если уже имеется, то использовать его), отправить исходные коды в удаленный репозиторий.
* 2.8. Склонировать удаленный репозиторий в папку, отличную от оригинальной.
* 2.9. Добиться конфликта файлов при применении изменений в оригинальном репозитории.
* 2.10. Разрешить конфликт.
* 2.11 Подготовить отчет о проделанной работе и прикрепить его в качестве ответа на это задание в learn. Отчет должен содержать список выполняемых в процессе выполнения лабораторной работы команд с их описанием, а также результат выполнения команд, скриншоты страниц с удаленного репозитория, чтоб были видны ветки и история операций (адрес репозитория также необходимо предоставить).

Примечание: так как до выполнения лабораторной работы был создан репозиторий, и туда же были добавлены файлы без использования консоли, то и работа будет выполняться на двух папках.



lab5\_repository – пустая папка

prof\_labs – папка с файлами, которая уже доступна в удалённом репозитории

**2.1 Создать локальный репозиторий**

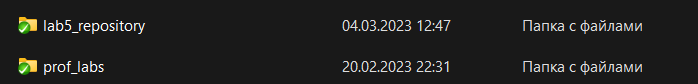
Создан локальный репозиторий в папке lab5\_repository с помощью git init.

Эта команда создаёт в текущем каталоге новый подкаталог с именем .git, содержащий все необходимые файлы репозитория — структуру Git репозитория. На этом этапе проект ещё не находится под версионным контролем.

D:\Repositories\lab5\_repository>git init

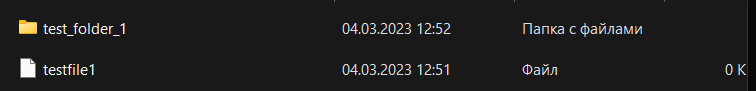
Initialized empty Git repository in D:/Repositories/lab5\_repository/.git/

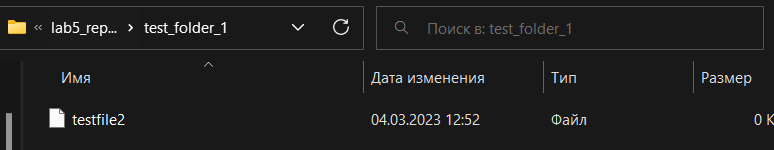
Теперь папка lab5\_repository имеет «галочку».



**2.2 Добавить несколько файлов**

Была создана папка test\_folder\_1, внутри которой есть пустой файл test\_file2. Ещё один пустой файл test\_file1 был создан рядом с папкой.





D:\Repositories\lab5\_repository>NUL> testfile1 – создаёт пустой файл

D:\Repositories\lab5\_repository>md test\_folder\_1 – создаёт пустую папку

Потом была скопирована домашняя работа (timestamp и минимальный элемент списка) из одной папки в другую, в lab5\_repository. Предварительно была для файлов создана папка.

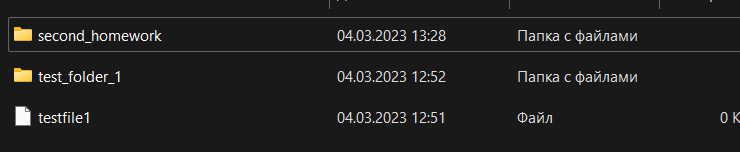
xcopy [исходная папка или файл] [папка назначения или файл] [параметры] - Эта команда копирует содержимое в папке в другую папку.

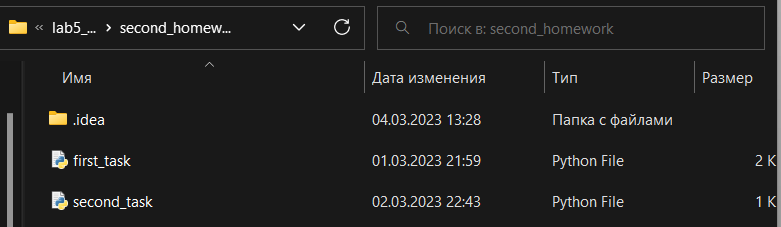
D:\> xcopy "D:\Задачи к предметам\prof\_labs\Домашние работы\Проекты домашней работы\second\_homework" "D:\Repositories\lab5\_repository\second\_homework" /e

…

…

Скопировано файлов: 9.





**2.3 Сделать один из файлов игнорируемым**

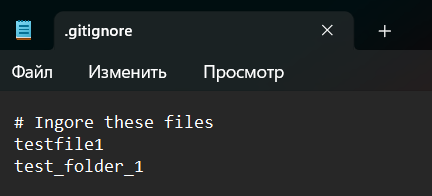
Для начала нужно создать файл .gitignore, в котором будут храниться названия файлом или шаблоны файлов, которые нужно игнорировать. Игнорировать эти файлы означает то, что их не нужно сохранять в удалённый репозиторий.

D:\Repositories\lab5\_repository>type nul > .gitignore

Затем я открыл этот файл для редактирования с помощью notepad (Это не совсем консольный редактор файла, он просто открывает файл).

D:\Repositories\lab5\_repository>notepad .gitignore

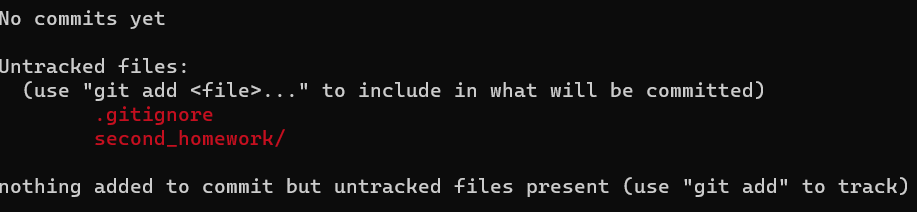
В этом файле было записано игнорирование ранее созданных пустых файлов:



Если игнорировать папку, то будут игнорированы и файлы в нём.

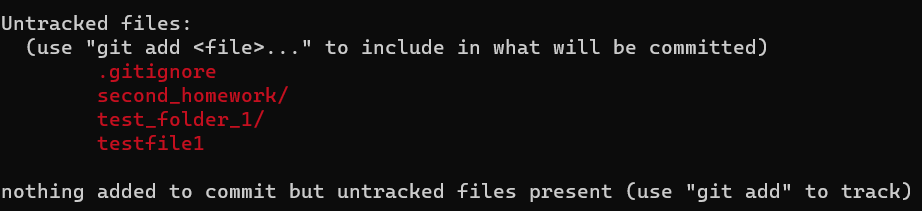
Чтобы проверить то, что эти файлы не будут сохраняться в удалённом репозитории, используется команда git status.

D:\Repositories\lab5\_repository>git status



То есть, сохранены будут только файл .gitignore и проект с домашней работой.

Если бы не было прописано игнорирование, то можно было бы зафиксировать и эти пустые файлы:

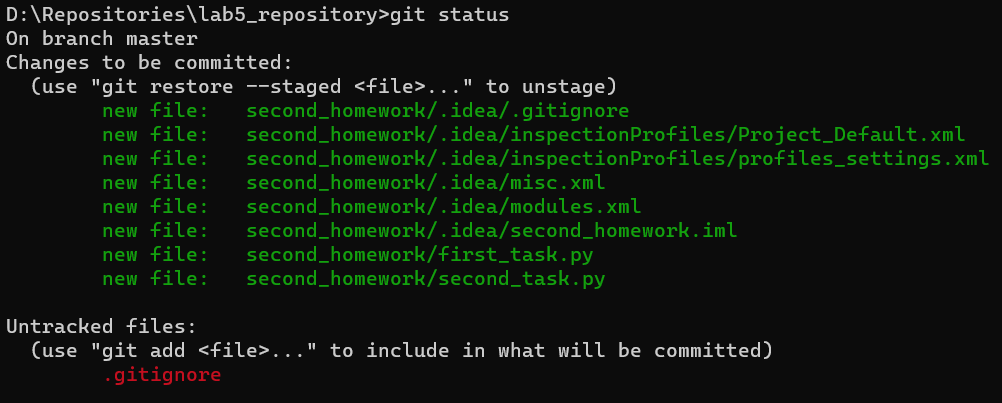


**2.4 Зафиксировать изменения**

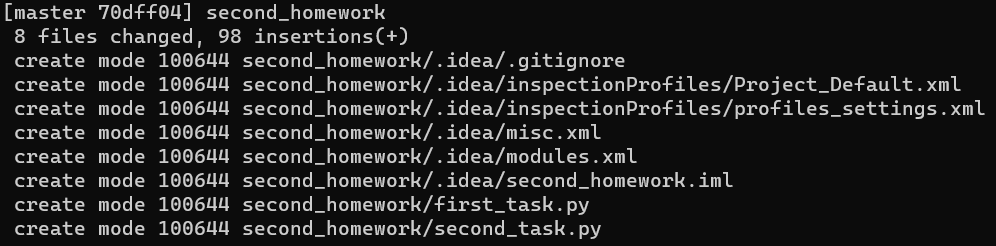
Для начала нужно добавить нужные файлы под версионный контроль. Когда git add используется как отдельная команда, она переносит ожидающие изменения из рабочего каталога в раздел проиндексированных файлов. git commit уже будет сохранять эти самые изменения.

D:\Repositories\lab5\_repository>git add second\_homework

В git status теперь они обозначаются так:

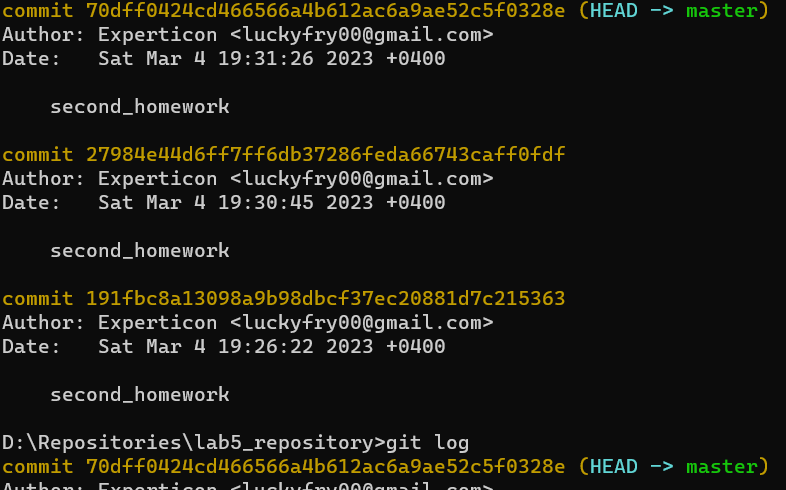


D:\Repositories\lab5\_repository>git commit -m second\_homework



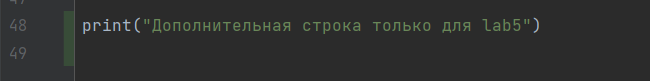
Командой git log можно просмотреть историю версий (Были проделаны некоторые «эксперименты» с commit).

D:\Repositories\lab5\_repository>git log

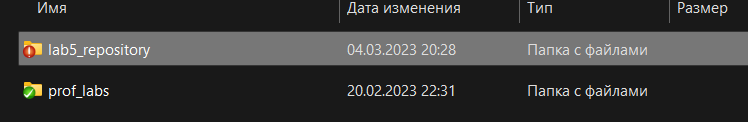


**2.5 Внести изменения в файлы и зафиксировать изменения**

Допустим, было выполнено изменение файла first\_task в проекте second\_homework:

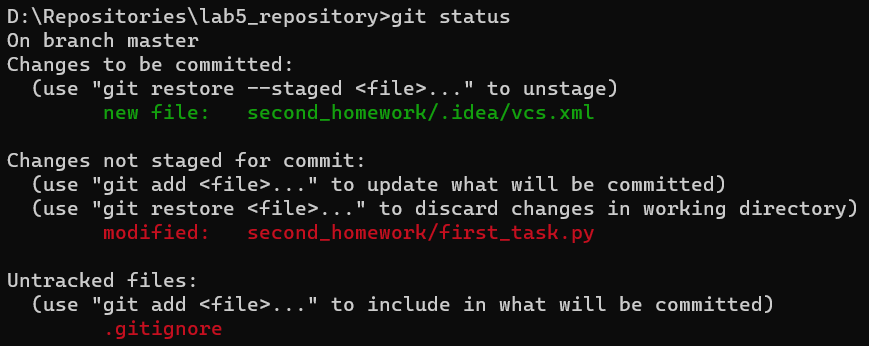


Теперь репозиторий имеет другой значок:



Тот же значок имеет и папка проекта, и сам файл.

При git status также есть изменения:

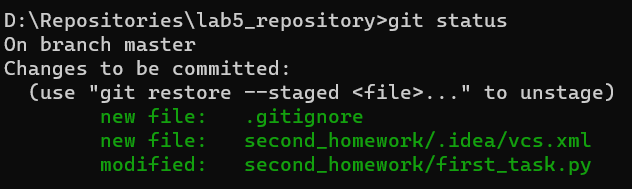


Далее были произведены эти действия:

* Добавлен .gitignore
* Добавлен изменённый файл first\_task.py
* Сохранены изменения

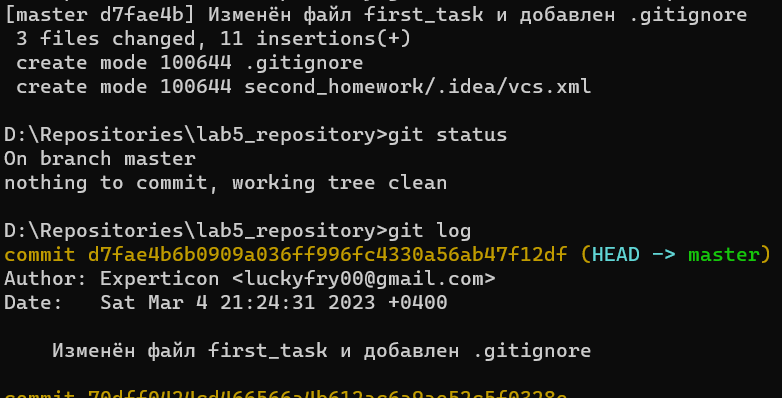
D:\Repositories\lab5\_repository>git add .gitignore

D:\Repositories\lab5\_repository>git add second\_homework/first\_task.py



D:\Repositories\lab5\_repository>git commit -m "Изменён файл first\_task и добавлен .gitignore"

Также проверено изменение с помощью git log.

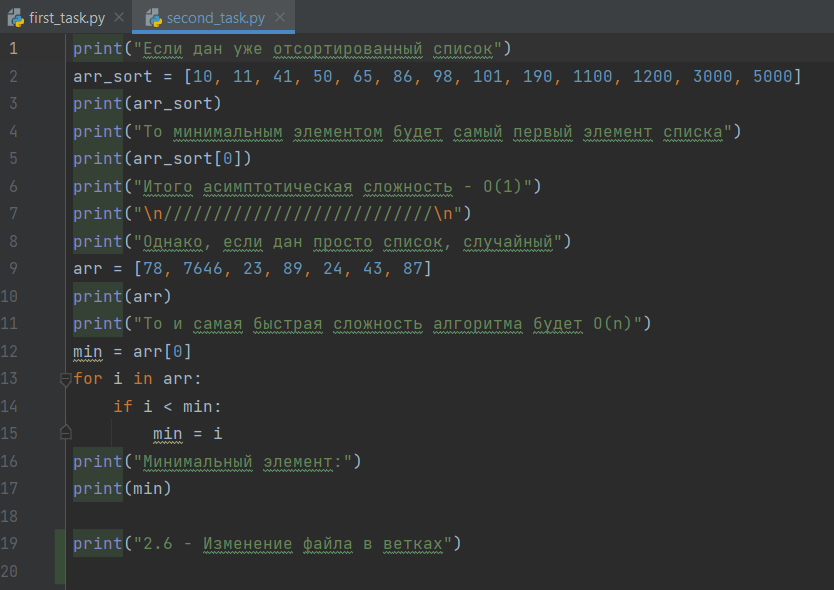


**2.6 Создать несколько бранчей, редактировать файлы в разных бранчах и фиксировать изменения. Осуществить слияние двух веток в одну.**

С помощью команды git checkout -b создаётся ветка и работа сразу же переходит к ней.

D:\Repositories\lab5\_repository>git checkout -b second\_task.py

Switched to a new branch 'second\_task.py'



В файле second\_task.py добавлена команда.

Далее идёт такой же коммит, как и ранее.

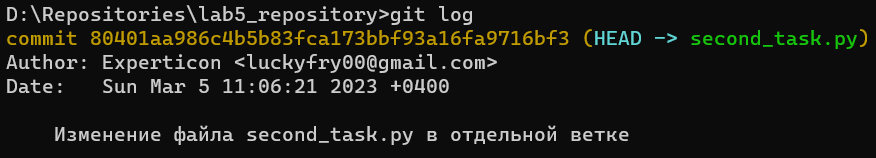
D:\Repositories\lab5\_repository>git add second\_homework/second\_task.py

D:\Repositories\lab5\_repository>git commit -m "Изменение файла second\_task.py в отдельной ветке"

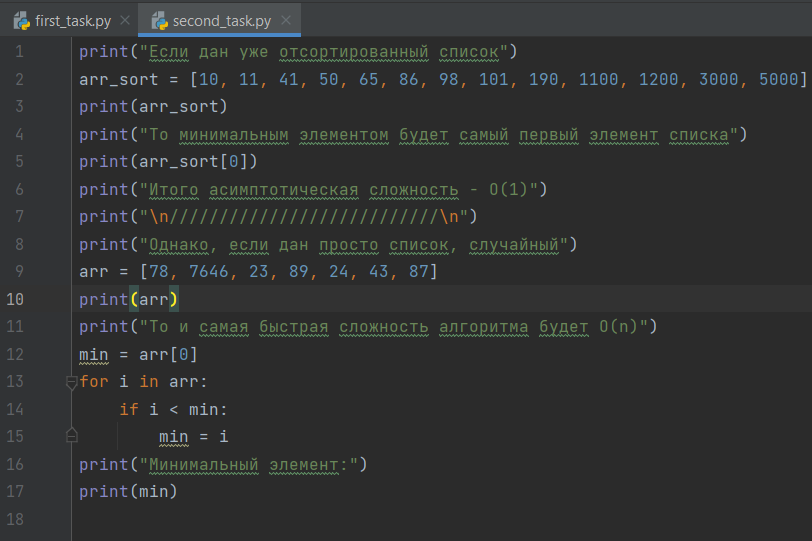
Далее можно все изменения добавлять с помощью комманды:

git commit -a -m – сразу добавляет изменения и сразу же их сохраняет.

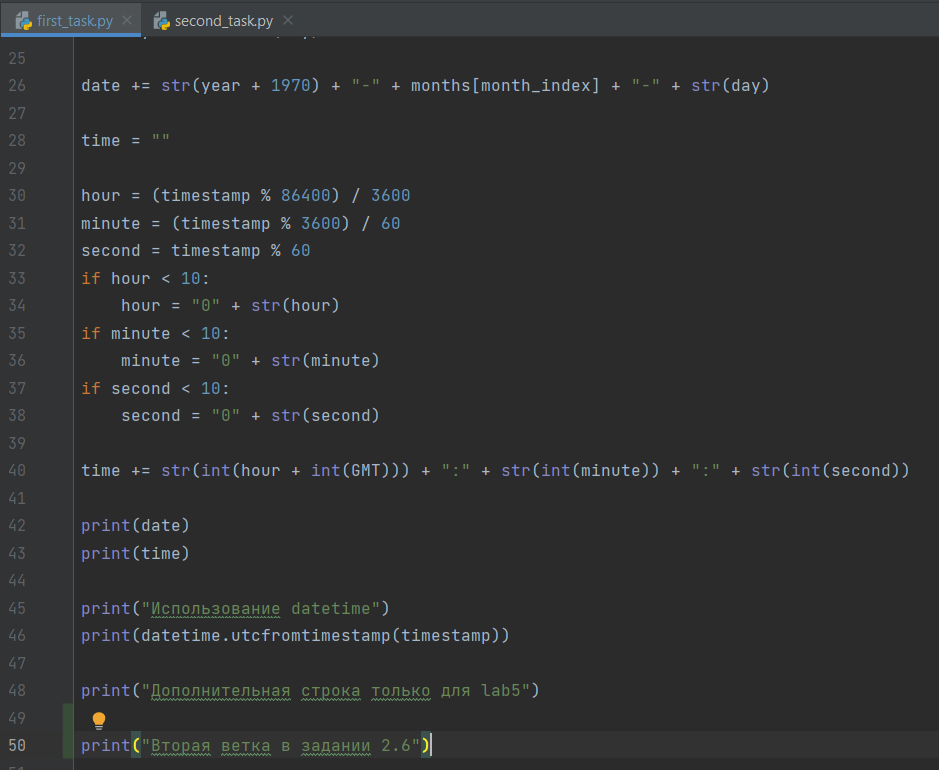
Результат:



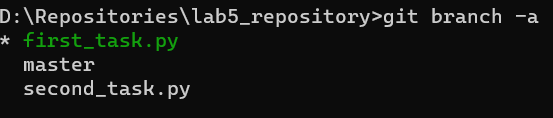
Если переключиться в ветку master, то там не будет этой строчки:



Подобные действия я выполнил и с другой веткой, файлом first\_task:



С помощью команды git branch -a можно увидеть все существующие на данный момент ветки:

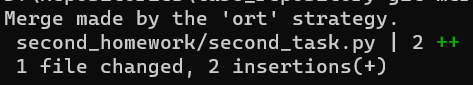


Теперь слияние веток. Для этого нужно переключиться на ту ветку, где слияние будет проводиться. Затем командой git merge слить с другой веткой.

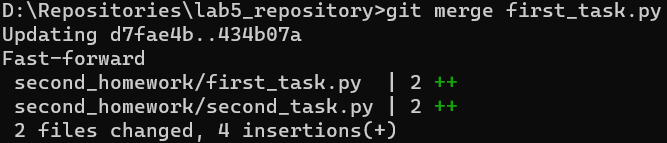
D:\Repositories\lab5\_repository>git checkout first\_task.py

Переход на ветку first\_task.py.

D:\Repositories\lab5\_repository>git merge second\_task.py – объединение с другой веткой.

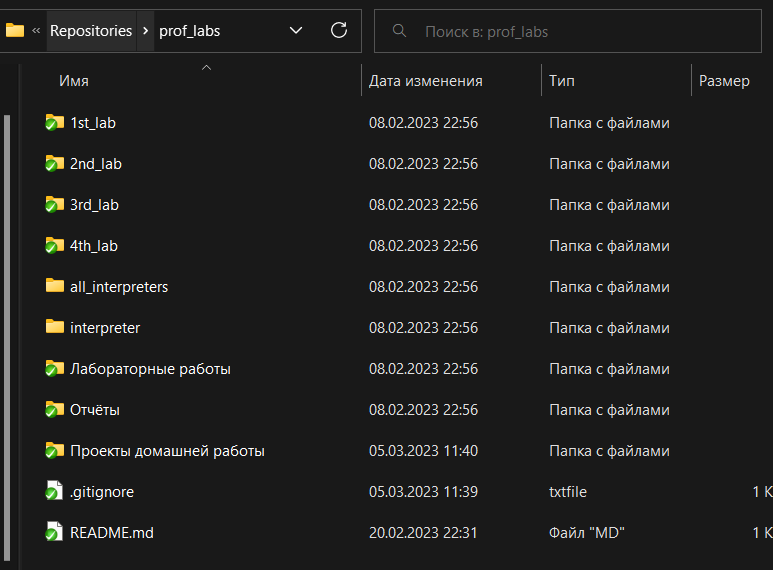


Теперь ветка first\_task.py имеет также изменения из другого файла. Ветка master всё ещё не имеет данных изменений. Проводится ещё одно слияние, только в основной ветке.

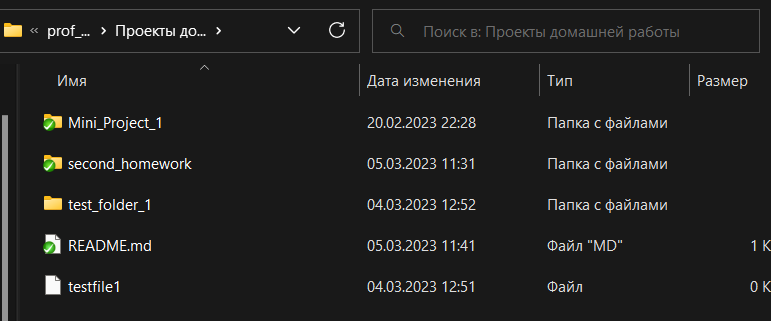


**2.7 Создать аккаунт github (если его нет, а если уже имеется, то использовать его), отправить исходные коды в удаленный репозиторий.**

Аккаунт GitHub уже существует. Теперь работа будет проводиться в репозитории prof\_labs в аккаунте “Experticon”. Проект second\_homework будет перенесён в prof\_labs. Файл .gitignore также был использован, чтобы не сохранять папки интерпретаторов Python (эксперименты с Python в начале).



Добавлен дополнительный README для проекта second\_homework и мусор (игнорируемые файлы с lab5\_repository).



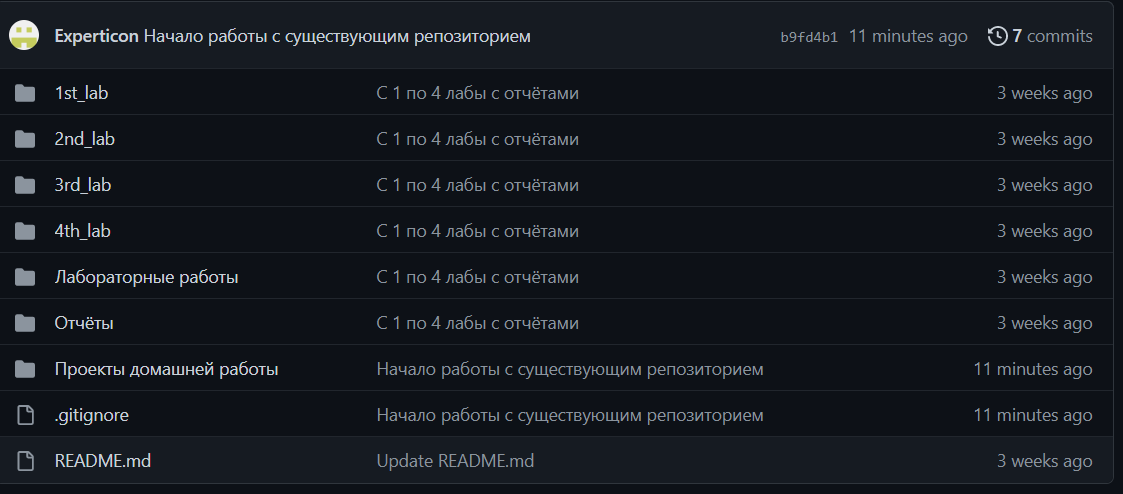
Всё также были использованы команды git add и git commit. Но теперь, чтобы ещё сохранить всё в удалённом репозитории, нужно было ввести ещё одну команду: git push.

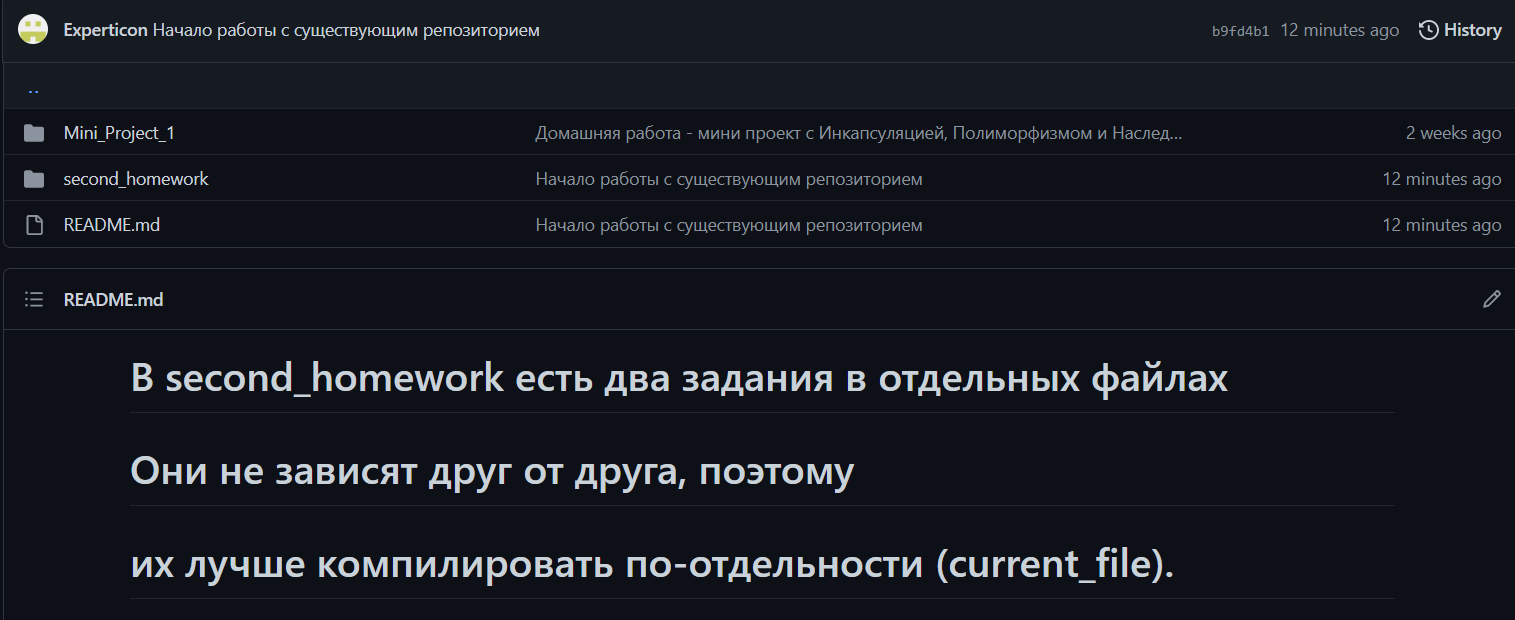
D:\Repositories\prof\_labs>git add –all

D:\Repositories\prof\_labs>git commit -m "Начало работы с существующим репозиторием"

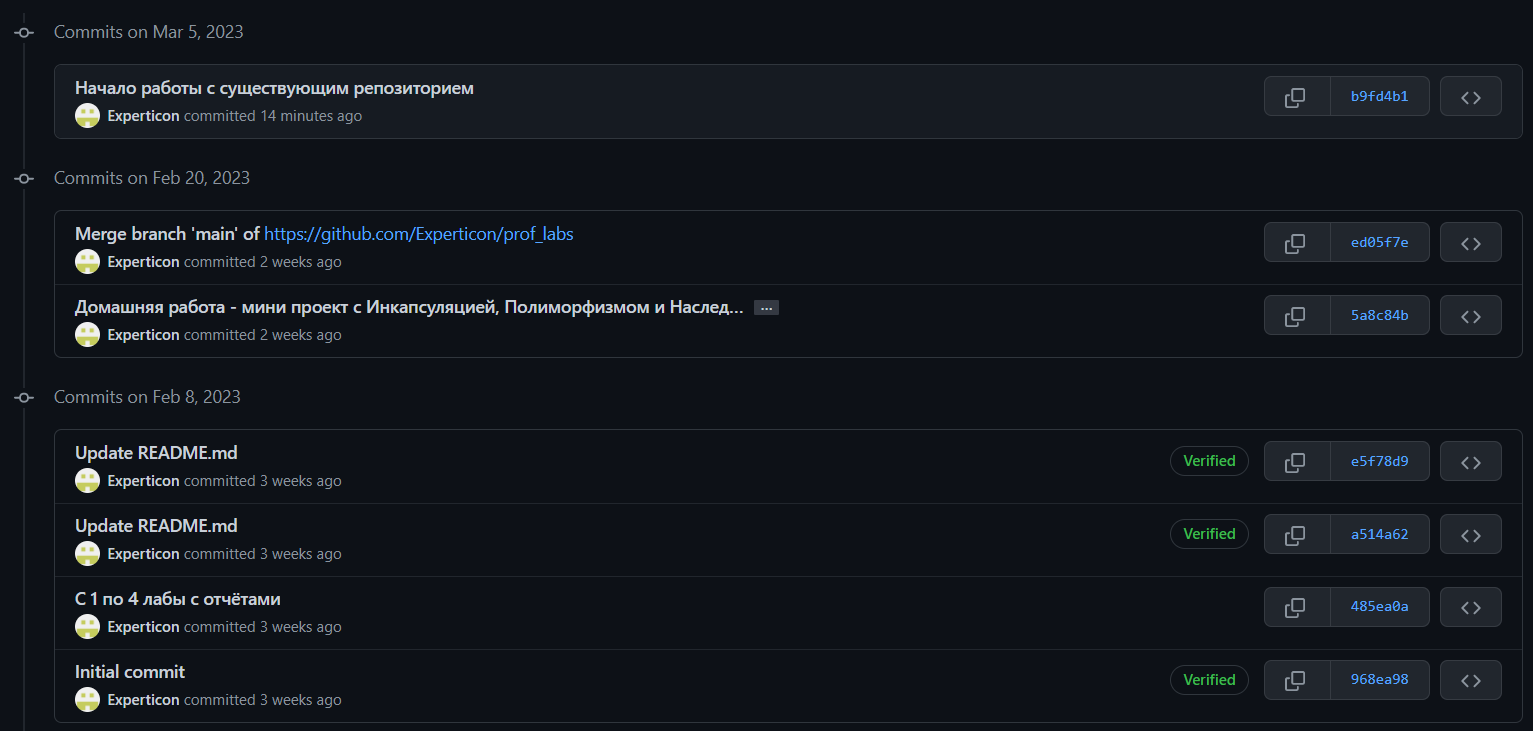
D:\Repositories\prof\_labs>git push https://github.com/Experticon/prof\_labs.git main

Результат в GitHub:



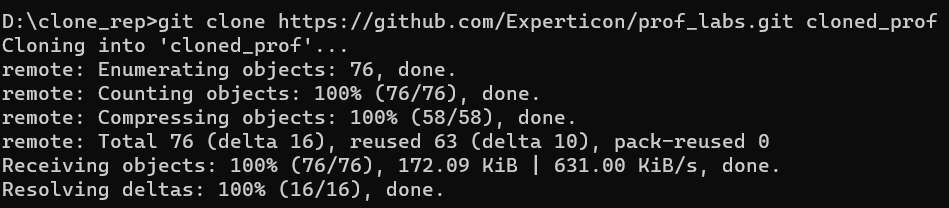


Всю историю коммитов также можно увидеть здесь:

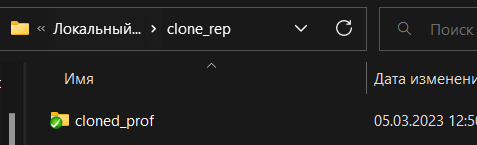


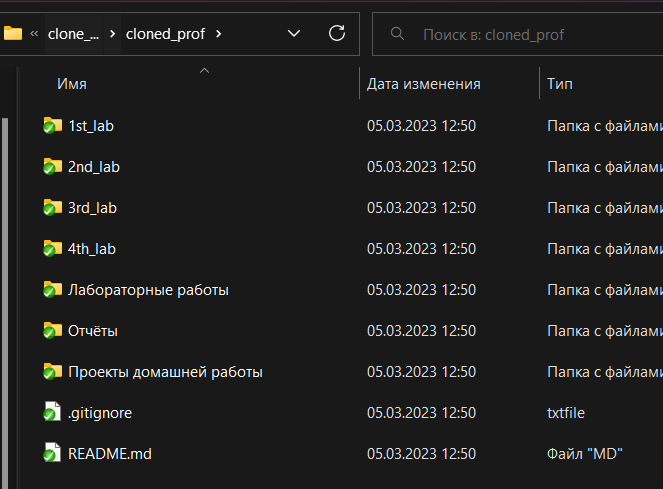
**2.8. Склонировать удаленный репозиторий в папку, отличную от оригинальной.**

Пусть будет создана некая папка clone\_rep на диске D. В него будет склонирован существующий удалённый репозиторий prof\_labs с именем cloned\_prof:



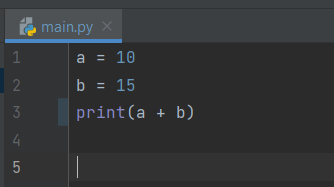
git clone делает то, что и говорит его название. Теперь можно работать в этой папке тоже.



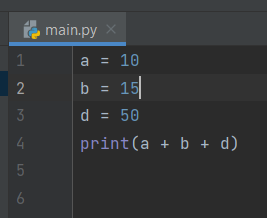


**2.9. Добиться конфликта файлов при применении изменений в оригинальном репозитории.**

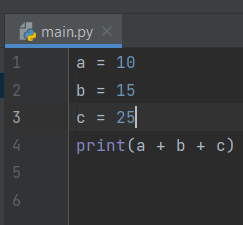
Пусть будет создана программа на Python, которая будет складывать переменные. Название проекта: conflict\_pr. Изначально содержимое файла такое. Именно в таком виде файл был сохранён.



Далее после процесса создания ветки, изменения и сохранения файлов вышло так, что в main ветке программа сохранена так:

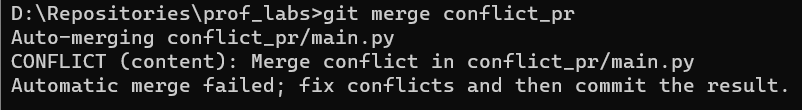


Когда в ветке conflict\_pr по-другому:

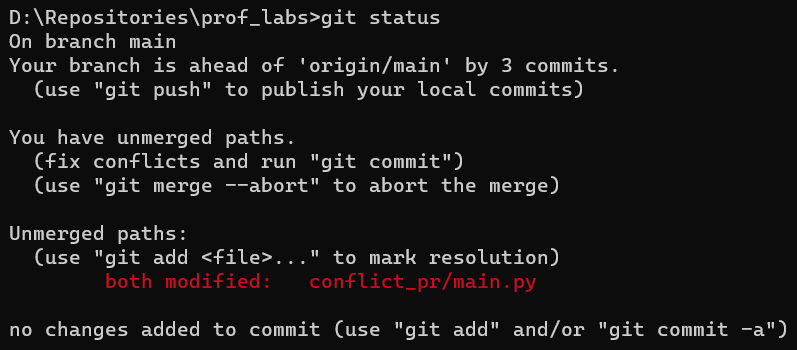


А именно, переменные другие. При попытке объединить две ветки (в ветке main с веткой conflict\_pr) всплывёт сообщение об конфликте:

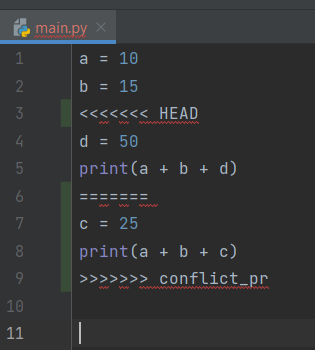
D:\Repositories\prof\_labs>git merge conflict\_pr



Более подробное описание конфликта можно узнать с помощью git status:



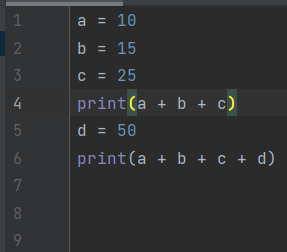
Так же в самой программе появились разделители веток:



**2.10. Разрешить конфликт.**

Самый простой способ разрешить конфликт — отредактировать конфликтующий файл, чтобы были совпдаения.

Было решено оставить программу именно такой:

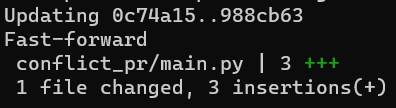


Здесь есть совпадения с conflict\_pr ветки, и особенность от main ветки. Этот результат был сохранён в main ветке. Тогда, чтобы все эти особенности перенести, нужно переключиться на ветку conflict\_pr и там сделать слияние:

D:\Repositories\prof\_labs>git checkout conflict\_pr

Switched to branch 'conflict\_pr'

D:\Repositories\prof\_labs>git merge main



Теперь ветки сохранены.

История коммитов (в первый раз не вышло разрешить конфликт).

