Отчет выполнил: студент ИВТ(ВМК)-21

Рычков Родион

**Система управления версий**

Git – это консольная утилита, для отслеживания и ведения истории изменения файлов, в вашем проекте. Чаще всего его используют для кода, но возможно и для других файлов. Например, для картинок – полезно для дизайнеров.

С помощью Git вы можете откатить свой проект до более старой версии, сравнивать, анализировать или сливать свои изменения в репозиторий.

Чтобы начать работать с Git, необходимо в проводнике Windows в контекстном меню выбрать «Git Bash Here»

Запустить Git лучшего всего в той папке, где у вас хранится исходный код программы.

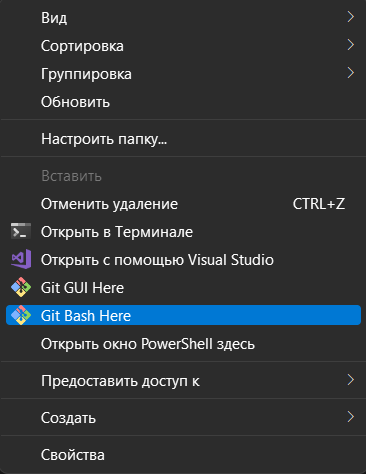


Рисунок 1 - Запуск Git'а

Необходимо создать репозиторий с помощью команды git init, который git создаст папку, где будет хранить информацию.



Рисунок – Создание репозитория

Создадим новую папку для работы с Git’ом

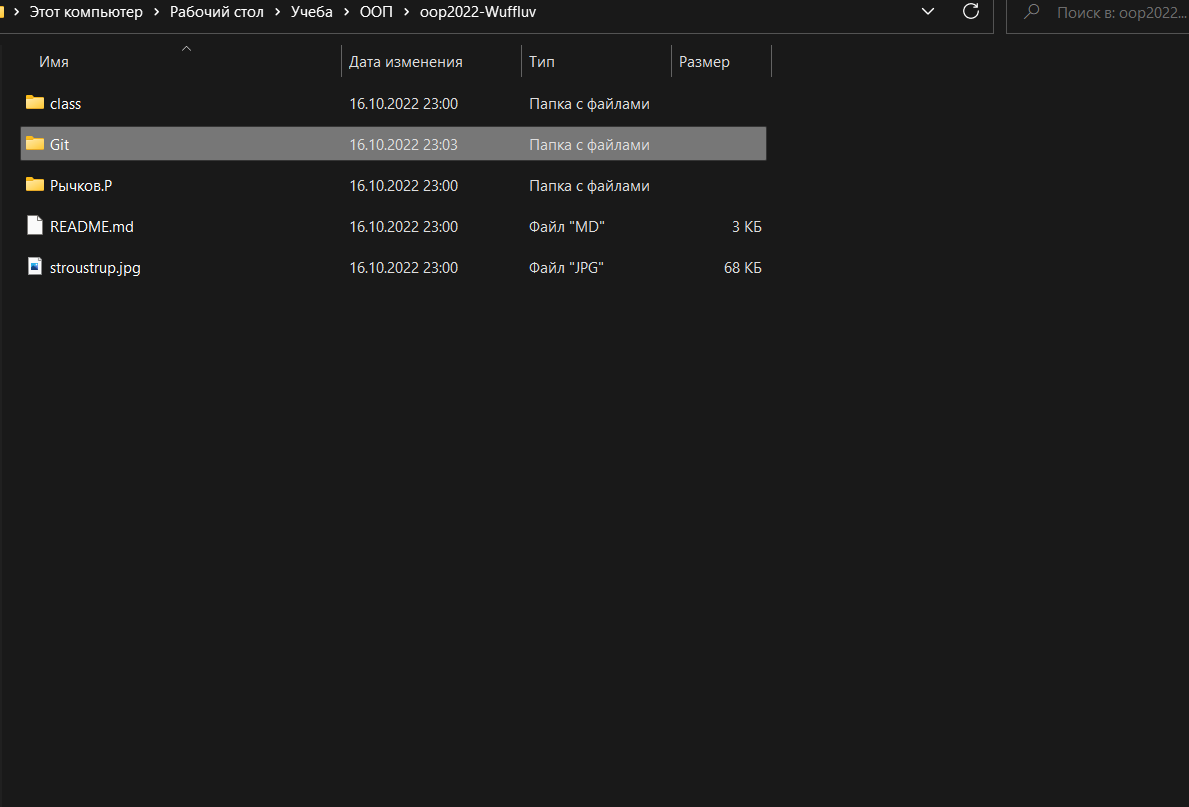


Рисунок - Создание новой папки для работы с Git'ом

Создав новую папку, создадим в VS новый файл для наглядной работы с версиями. Так как я уже создал папку и сам файл, нужно проверить в Git’е статус репозитория. Для этого пропишем команду git status

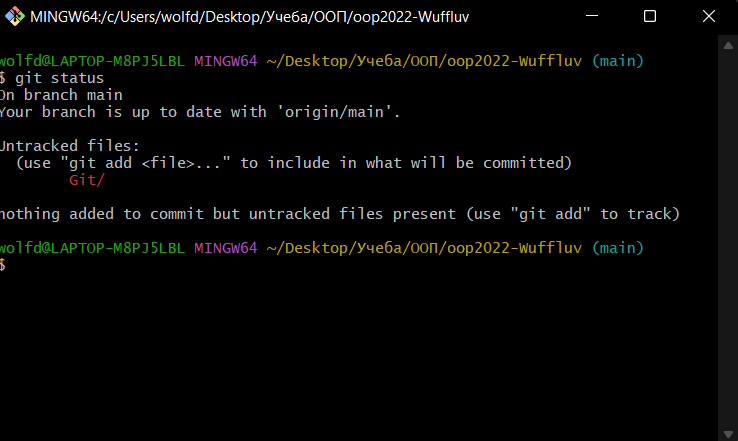


Рисунок - проверка статуса репозитория

Мы видим, что Git уведомляет нас о том, что в репозиторий был добавлен новый файл, а точнее папка, в которой у нас находится программа. Необходимо добавить эту папку в контроль Git’у. Для этого необходимо прописать команду git add . Данная команда с точкой в конце, позволит добавить нам все файлы, это удобно т.к не надо прописывать каждый файл.

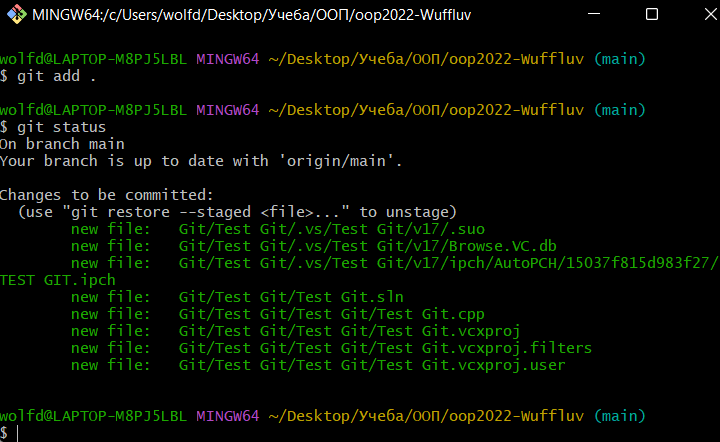


Рисунок - Добавление новых файлов

Приступим к первым изменениям. Добавим в программе вывод надписи “Hello Git!”

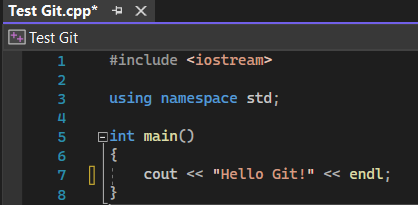


Рисунок - добавление первых изменений

Проверим с помощью git status наши изменения

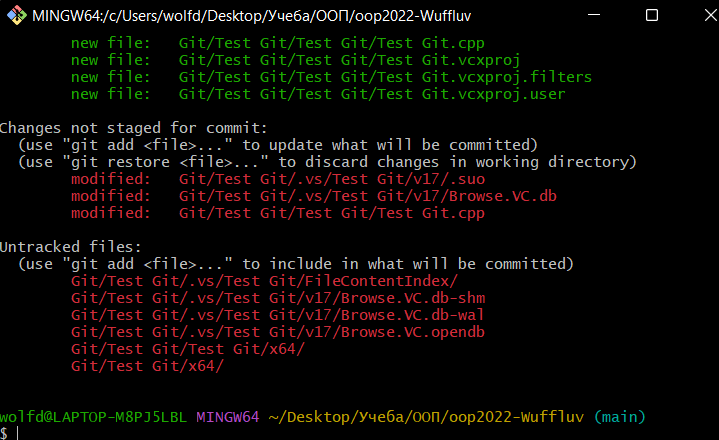


Рисунок - Изменения в репозитории

Как мы видим, Git заметил изменения в нашей программе. Необходимо прописать команду git add для того, чтобы Git зафиксировал данные изменения. После работы с файлом, необходимо сохранить изменения командой git commit -m “имя коммита”.

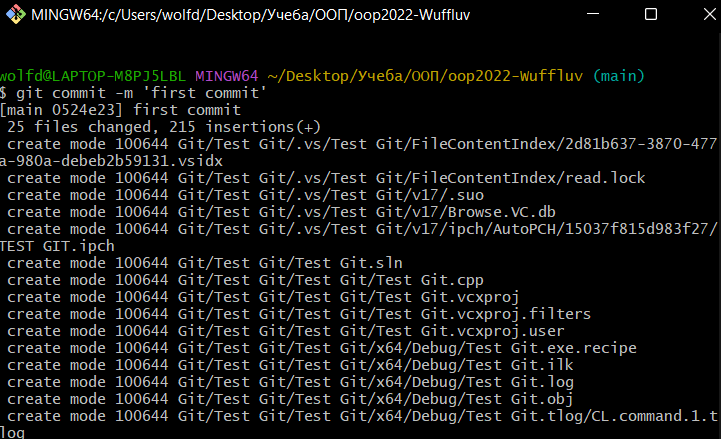


Рисунок - Первый коммит

Мы создали первый коммит! Приступим к главной особенности Git’а – это его контроль версий. Существуют «ветки», по которым мы сможем отслеживать различные версии нашей программы. В ветке main, в которой мы сейчас находимся, наша программа имеет лишь один вывод – Hello Git. Приступим к созданию новой версии нашей программы. Необходимо прописать команду git branch “имя ветки”.

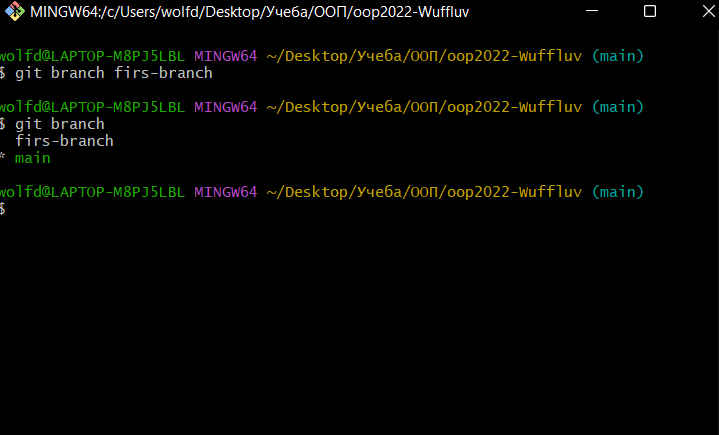


Рисунок - Создание новой ветки приложения

Как мы видим, git создал новую ветку под названием firs-branch. Чтобы перейти к ней и внести изменения в наш код, необходимо прописать команду git checkout “имя необходимой ветки”.



Рисунок - Переключение на другую ветку

Внесем в наш код изменения, добавим новый вывод предложения: “Git is free and open source software for distributed version control”.

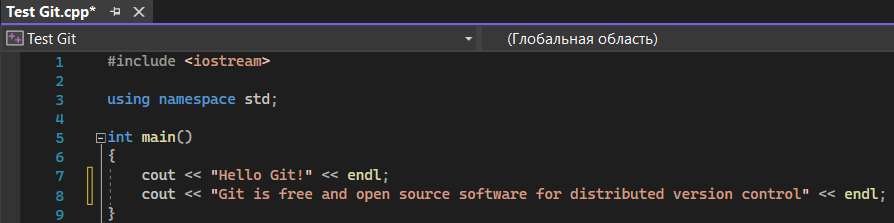


Рисунок - Добавление новой строки кода

После внесения данного изменения, необходимо прописать git add . для добавления файлов в Git. Также необходимо прописать команду git commit -m “имя коммита”.

Пришло время проверки наших версий приложения. Как вы помните в ветке main в нашем коде была только одна строка с выводом Hello Git, а в ветке firs-branch мы добавили новую строку вывода. Для сравнения версий, перейдем в ветку main, а потом в ветку firs-branch и сравним код. Для этого пропишем команду git checkout main – для перехода в ветку main.

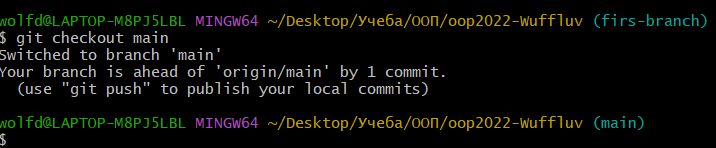


Рисунок - Переход в ветку main

Откроем код нашей программы.

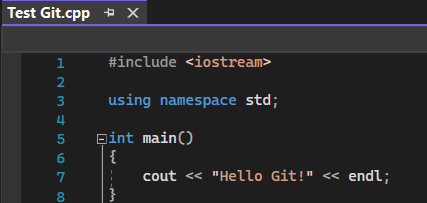


Рисунок - Код в ветке main

Фиксируем, что в ветке main всего лишь одна строка. Перейдем в ветку firs-branch и откроем код нашей программы.

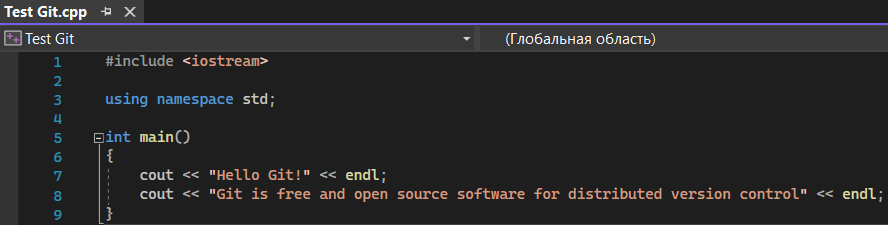


Рисунок - код в ветке firs-branch

Как мы видим, в разных версиях, разный код. В этом и заключается особенность Git’a. Благодаря этому мы можем контролировать версии нашей программы, главное не забывать их фиксировать.

Чтобы добавить новые изменения в ветку main, необходимо совместить наши ветки, для этого пропишем команду git merge “имя ветки, с которой необходимо взять изменения”.

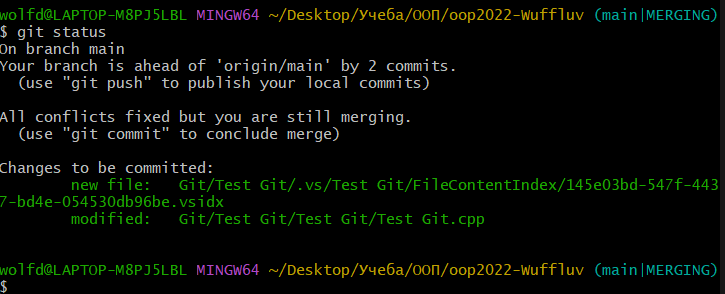


Рисунок - совмещение веток в одну

Совместив данные ветки, наш код программы будет иметь уже две строки вывода, находясь в ветке main.

Необходимо создать файл .gitignore и указать в нем файлы, за которыми Git’у следить не нужно, например это, временные и исполняемые файлы. Так, мы сильно уменьшим размер репозитория.

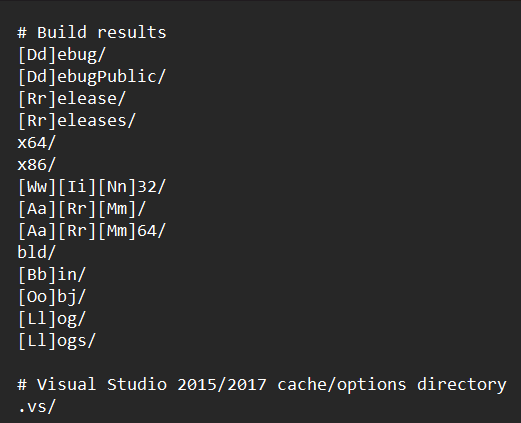


Рисунок - файл .gitignore

После этого необходимо добавить данный файл Git’у, используя команду git add .

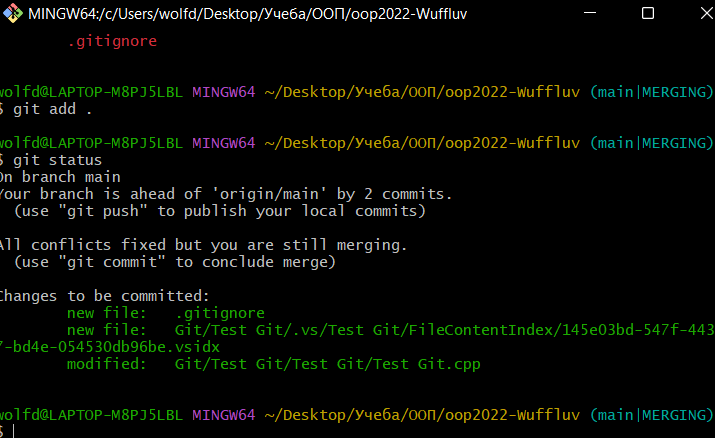


Рисунок - добавление .gitignore

После этого необходимо «закоммитить» состояние данного репозитория.

Для отправки ветки main в удаленный уже созданный репозиторий:

git remote add origin <https://github.com/Username/Reposname.git>

git push [имя удаленного сервера] [ветка]

git push -u origin main

origin – псевдоним для удаленного репозитория

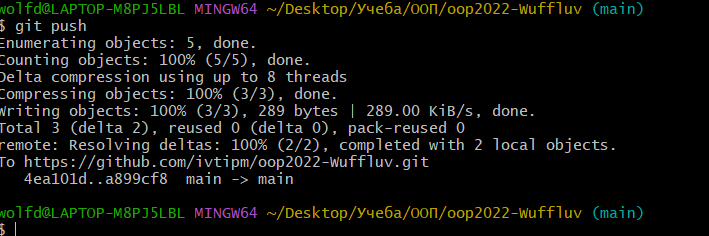


Рисунок - отправка изменений

После добавления изменений в код программы через GitHub, мы можем забрать изменения из удаленного репозитория командой git pull “имя удаленного репозитория” “имя ветки удаленного репозитория”. Тогда ваш код примет вид, как в удаленном репозитории.

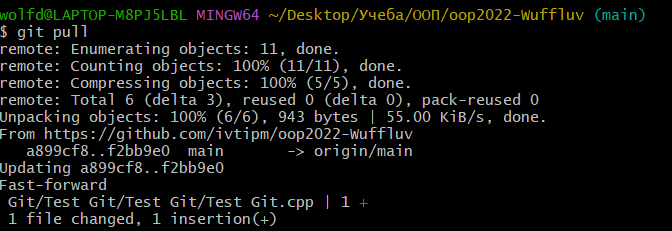


Рисунок - скачивание изменений из удаленного репозитория