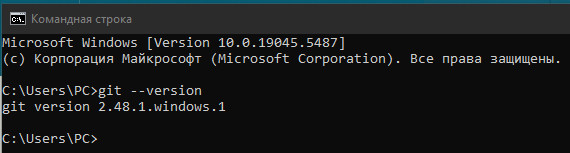
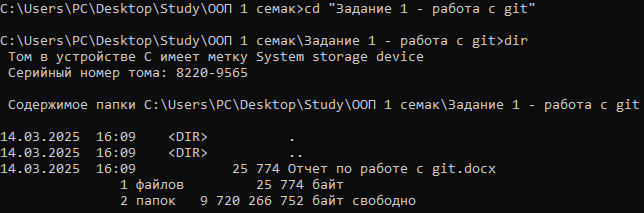
Отчет выполнил: студент ВМК-22, Александров Данил Сергеевич  
  
Git – система управления версий

Git – это технология, созданная для контроля версий.

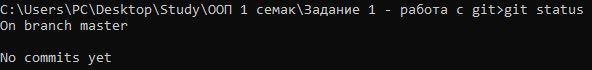
Для взаимодействия с git требуется какой-либо терминал. В данном будет использоваться Командная строка Windows 10.

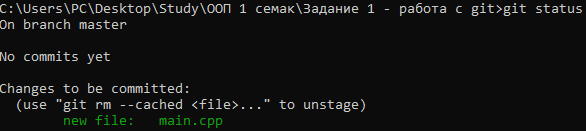
Инициализация проекта

git --version – команда для просмотра текущей версии установленного git.  
  
рис. 1 – просмотр установленной версии.

Для того чтобы связать какой-либо проект с git необходимо в консоли перейти в папку проекта.  
dir – команда просмотра текущей директории.   
cd – команда смена директории.  
cd .. – команда для выхода к директории проекта.  
mkdir <name of directory> - создание новой директории  
  
рис. 2 – переход в папку проекта

git init – команда для инициализации репозитория   
  
рис. 3 – инициализация выбранного репозитория   
В результате в папке проекта будет создана системная папка «.git»

git status – команда для проверки статуса  
  
рис. 4 – просмотр статуса.

git add <file> – команда для добавления файла в систему контроля версий для слежки за ним.  
git rm --cached <file> - команда прекращения отслеживания git’ом файла  
git add . – команда добавления всех файлов в репозитории в git  
  
рис. 5 – добавление файла в git.

Авторизация пользователя в git

Без авторизации пользователя нельзя оставлять коммиты, потому что Git не знает, кто вы, то есть у вас не настроены имя пользователя и email, которые необходимы для создания коммитов. Git требует эту информацию, чтобы привязать ваши изменения к конкретному автору.

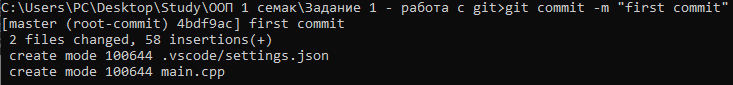
Этапы авторизации:

1. Укажите email - git config --global user.email "<email>"
2. Укажите имя - git config --global user.name "<name >"

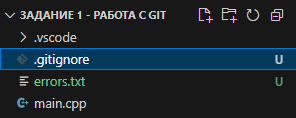
Коммит

Коммит — это сохранение изменений в Git, позволяющее зафиксировать текущее состояние проекта.

git commit -m “<name of commit>” – добавление коммита  
Опция -m позволяет сразу указать сообщение коммита в самой команде, в кавычках.

  
рис.6 – добавление коммита.  
4bdf9ac – уникальный код коммита для последующего возвращения к нему(к этому этапу разработки приложения)

Файл .gitignore

Файл .gitignore создается для того, чтобы в нем находилось то, что не следует отслеживать системой git. В нем мы просто перечисляем названия таких файлов.  
  
рис. 7 – создание gitignor

Работа с ветками

Ветки в Git нужны, чтобы:

* Работать над разными задачами параллельно.
* Сохранять основной код стабильным.
* Экспериментировать без риска.
* Упрощать командную работу.
* Управлять версиями проекта.

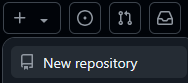
Изначально мы находимся в ветке «master».  
git branch – просмотр доступных веток.

git branch <name of branch> - создание новой ветки.  
git branch -D <name of branch> - удаление ветки.  
git checkout <name of branch> - переключение на выбранную ветку  
git checkout -b <name of branch> - создание + переключение на ветку  
Опция -b позволяет сразу переходить на созданную ветку.  
git branch -M main – переименование текущей ветки

git merge <name of needed branch> - соединение выбранной ветки с текущей.

Работа с GitHUB

Шаги публикации локального содержимого на GitHUB:

1. Создание нового репозитория  
   
2. Привязка удаленного репозитория к нашему локальному  
   git remote add origin https://github.com/nesquikprogs/OOP\_1.git
3. Загрузка локального содержимого в GitHUB  
   git push -u origin <selected branch>  
   git push — это команда для отправки/публикации изменений/коммитов из локального репозитория на компьютере в удалённый репозиторий(на GitHUB)

origin — это стандартное имя удалённого репозитория, к которому привязан локальный репозиторий  
-u — это сокращение от --set-upstream. Она используется, чтобы связать локальную ветку с соответствующей веткой в удалённом репозитории

git push – публикация на GitHUB созданного коммита.  
git clone <url GitHUBrepos> - копировать в текущую выбранную папку репозиторий с GitHUB.  
git pull – загрузка на компьютер данных из репозитория на GitHUBs