

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ
АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО
ОБРАЗОВАНИЯ «СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ
УНИВЕРСИТЕТ»**

ИНСТИТУТ ЦИФРОВОГО РАЗВИТИЯ

**Отчёт о лабораторной работе №1 по дисциплине основы программной
инженерии**

Выполнила:

Нестеренко Тамара Антоновна,
2 курс, группа ПИЖ-б-о-20-1,

Проверил:

Доцент кафедры инфокоммуникаций,
Воронкин Р.А.

Ставрополь, 2021 г.

ВЫПОЛНЕНИЕ

1. Работа с консолью Git

1.1 Отображение текущей версии Git

```
C:\Users\тома нестеренко>git version
git version 2.33.0.windows.2
```

Рисунок 1 – Результат выполнения команды «git version»

1.2 Внесение изменений в локальный репозиторий и отправка на сервер

```
C:\Users\тома нестеренко>git clone https://github.com/tamaranesterenko/helloworld.git
Cloning into 'helloworld'...
remote: Enumerating objects: 34, done.
remote: Counting objects: 100% (34/34), done.
remote: Compressing objects: 100% (20/20), done.
remote: Total 34 (delta 11), reused 20 (delta 7), pack-reused 0
Receiving objects: 100% (34/34), 5.18 KiB | 442.00 KiB/s, done.
Resolving deltas: 100% (11/11), done.
```

Рисунок 2 – Результат клонирования репозитория на компьютер

Папка клонированного репозитория появилась на диске C. Результат представлен на рисунке 3.

Этот компьютер > Acer (C:) > Пользователи > тома нестеренко > helloworld

Имя	Дата изменения	Тип	Размер
.gitignore	09.09.2021 11:52	Текстовый докум...	1 КБ
Hello.txt	09.09.2021 11:52	Текстовый докум...	1 КБ
LICENSE	09.09.2021 11:52	Файл	2 КБ
README.md	09.09.2021 11:52	Файл "MD"	1 КБ

Рисунок 3 – Результат отображения, клонированного репозитория на компьютере

```
C:\Users\тома нестеренко>cd helloworld
C:\Users\тома нестеренко\helloworld>
```

Рисунок 4 – Результат открытия каталога хранилища

```
C:\Users\тома нестепенко\helloworld>git status
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean
```

Рисунок 5 – Результат работы команды «git status», проверка состояния репозитория

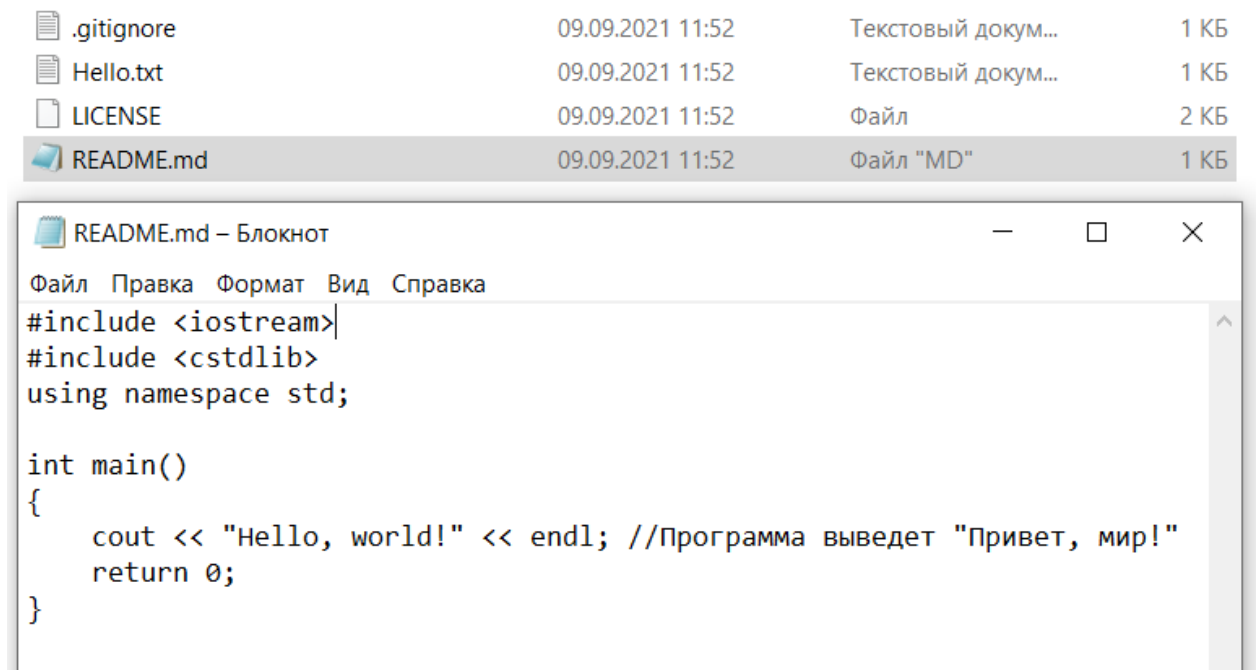


Рисунок 6 – Пример текста файла README.md до внесения изменений

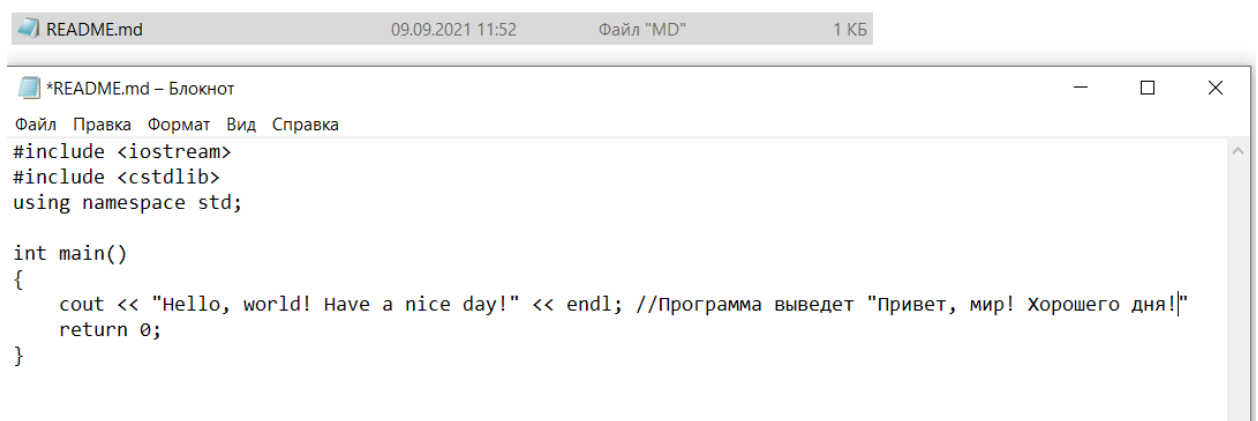


Рисунок 7 – Пример текста файла README.md после внесения изменений

```
C:\Users\тома нестеренко\helloworld>git add README.md
```

Рисунок 8 – Пример добавления файла в версионный контроль

```
C:\Users\тома нестеренко\helloworld>git commit -m "Changed file README.md"
[main e653afa] Changed file README.md
1 file changed, 1 insertion(+), 1 deletion(-)
```

Рисунок 8 – Пример фиксации изменений в файле в версионном контроле

```
C:\Users\тома нестеренко\helloworld>git push
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 4 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 352 bytes | 352.00 KiB/s, done.
Total 3 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To https://github.com/tamaranesterenko/helloworld.git
05b9662..e653afa main -> main
```

Рисунок 9 – Процесс загрузки файлов на удалённый сервер

main ▾


1 branch





0 tags


Go to file

Add file ▾

Code ▾

 **tamaranesterenko** Added the student's data bdeb1d6 now 🕒 14 commits

 .gitignore	Initial commit	6 days ago
 Hello.txt	Changed file Hello.txt	2 hours ago
 LICENSE	Initial commit	6 days ago
 README.md	Added the student's data	now

README.md 

Работу выполнила Нестеренко Тамара, студентка 2 курса группы ПИЖ-6-о-20-1

#include #include using namespace std;

int main() { cout << "Hello, world! Have a nice day!" << endl; //Программа выведет "Привет, мир! Хорошего дня!"

return 0; }

Рисунок 10 – Пример отображения списка файлов в удалённом репозитории

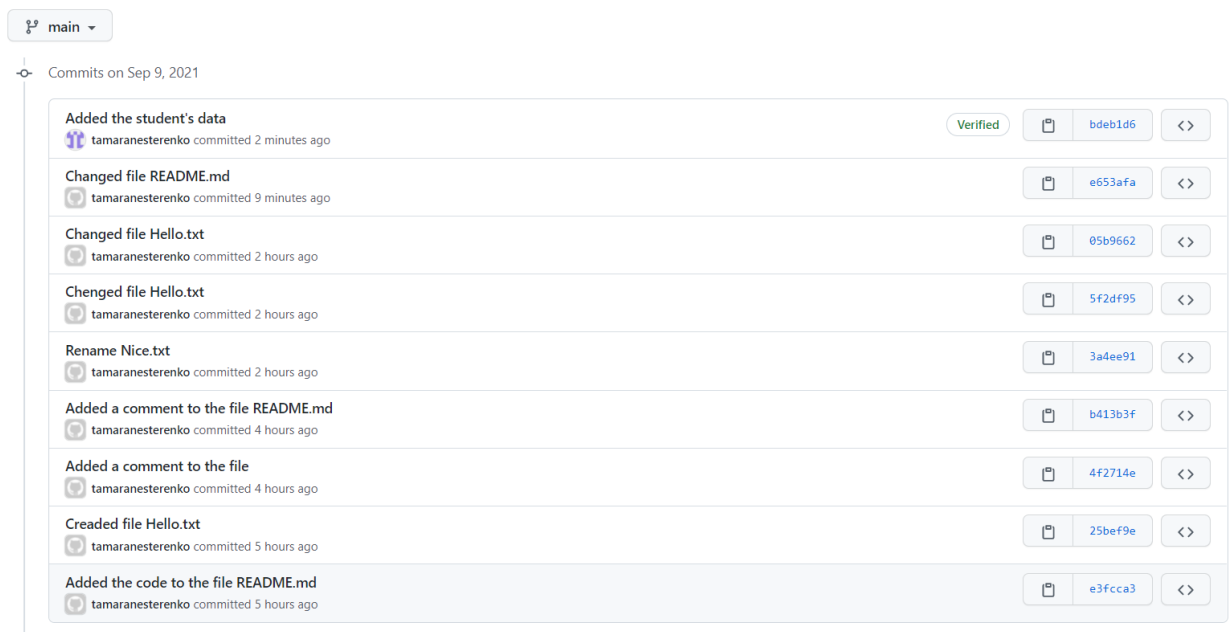


Рисунок 11 – Пример списка загруженных версий исходного файла в удалённом репозитории

2. Ответы на вопросы

1. Что такое СКВ и каково её назначение? Система контроля версий (СКВ) — это система, регистрирующая изменения в одном или нескольких файлах с тем, чтобы в дальнейшем была возможность вернуться к определённым старым версиям этих файлов.

2. В чем недостатки локальных и централизованных СКВ? Проблемой локальной СКВ является основное свойство — локальность. Она совершенно не предназначена для коллективного использования.

Самый очевидный минус централизованной СКВ - это единая точка отказа, представленная централизованным сервером. Если этот сервер выйдет из строя на час, то в течение этого времени никто не сможет использовать контроль версий для сохранения изменений.

3. К какой СКВ относится Git? Git — распределённая система контроля версий.

4. В чем концептуальное отличие Git от других СКВ? Основное отличие Git от любой другой СКВ — это подход к работе со своими данными. Подход Git к хранению данных больше похож на набор снимков миниатюрной файловой системы. Каждый коммит сохраняет состояние своего проекта в Git, система запоминает, как выглядит каждый файл в этот момент, и сохраняет ссылку на этот снимок.

5. Как обеспечивается целостность хранимых данных в Git? В Git для всего вычисляется хеш-сумма, и только потом происходит сохранение. В дальнейшем обращение к сохранённым объектам происходит по этой хеш-сумме. Это значит, что невозможно изменить содержимое файла или директории так, чтобы Git не узнал об этом. Данная функциональность встроена в Git на низком уровне и является неотъемлемой частью его философии. Нельзя потерять информацию во время её передачи и не получить повреждённый файл без ведома Git. Механизм, которым пользуется Git при вычислении хеш-сумм, называется SHA-1 хеш. Это строка длиной в 40 шестнадцатеричных символов (0-9 и a-f), она вычисляется на основе содержимого файла или структуры каталога.

6. В каких состояниях могут находиться файлы в Git? Как связаны эти состояния? У Git есть три основных состояния, в которых могут находиться файлы: зафиксированное (committed), изменённое (modified) и подготовленное (staged). Зафиксированный значит, что файл уже сохранён в вашей локальной базе. К изменённым относятся файлы, которые поменялись, но ещё не были зафиксированы. Подготовленные файлы — это изменённые файлы, отмеченные для включения в следующий коммит.

Три основных секции проекта Git: Git-директория (Git directory), рабочая директория (working directory) и область подготовленных файлов (staging area). Git-директория — это то место, где Git хранит метаданные и базу объектов проекта. Это самая важная часть Git, и это та часть, которая копируется при клонировании репозитория с другого компьютера. Рабочая директория является снимком версии проекта. Файлы распаковываются из сжатой базы данных в Git-директории и располагаются на диске, для того чтобы их можно было изменять и использовать. Область подготовленных файлов — это файл, обычно располагающийся в Git-директории, в нём содержится информация о том, какие изменения попадут в следующий коммит. Эту область ещё называют “индекс”, однако называть её stage-область также общепринято.

7. Что такое профиль пользователя в GitHub? Профиль - это публичная страница на GitHub, как и в социальных сетях. Работодатели могут посмотреть профиль GitHub и принять его во внимание, когда будут решать, брать на работу или нет.

8. Какие бывают репозитории в GitHub? Общедоступные и приватные.

9. Укажите основные этапы модели работы с GitHub.

1) Создание аккаунта. 2) Создание репозитория. 3) Клонирование репозитория. 4) Локальное изменение содержимого.

10. Как осуществляется первоначальная настройка Git после установки? Для этого необходимо перейти на официальный сайт Git <https://git-scm.com/> и

скачать версию для операционной системы. После чего необходимо выполнить установку на свой компьютер. Чтобы убедиться, что Git был успешно установлен, нужно ввести команду `git version` ниже в терминале, чтобы отобразить текущую версию Git.

11. Опишите этапы создания репозитория в GitHub.

- Имя репозитория может быть любое, необязательно уникальное во всем github, потому что привязано к аккаунту, но уникальное в рамках тех репозиториях, которые создавали.
- Описание (Description). Можно оставить пустым.
- Public/private. Выбираем открытый (Public), НЕ ставим галочку “Initialize this repository with a README”.
- gitignore можно сейчас не выбирать. Лицензия (License) – MIT.
- После заполнения этих полей нажимаем кнопку Create repository.

12. Какие типы лицензий поддерживаются GitHub при создании репозитория? Apache License 2.0, GNU General Public License v3.0, MIT License, BSD 2-Clause “Simplified” License, BSD 3-Clause “New” or “Revised” License, Boot Software License 1.0 и т.д.

13. Как осуществляется клонирование репозитория GitHub? Зачем нужно клонировать репозиторий? После создания репозитория его необходимо клонировать на компьютер. Клонирование выполняется для несения в репозиторий локальных изменений запроса на обновление файлов в удалённом репозитории.

14. Как проверить состояние локального репозитория Git?

Необходимо ввести команду `git status`.

15. Как изменяется состояние локального репозитория Git после выполнения следующих операций: добавления/изменения файла в локальный репозиторий Git; добавления нового/измененного файла под версионный контроль с помощью команды `git add`; фиксации (коммита) изменений с помощью команды `git commit` и отправки изменений на сервер с помощью команды `git push`? При добавлении/изменении файлов они помечаются как «modified». При добавлении под версионный контроль файл помечается как «new file». При фиксации изменений статус меняется на «... ahead on 1 commit», после отправки статус выдает: «your branch is up to date».

В локальной папке репозитория появится новый файл.

16. У Вас имеется репозиторий на GitHub и два рабочих компьютера, с помощью которых Вы можете осуществлять работу над некоторым проектом с использованием этого репозитория. Опишите последовательность команд, с помощью которых оба локальных репозитория, связанных с репозиторием

GitHub будут находиться в синхронизированном состоянии. **Примечание:** описание необходимо начать с команды **git clone**. Необходимо клонировать исходный репозиторий на каждый из компьютеров с помощью команды «**git clone**», на каждом компьютере после внесения локальных изменений нужно добавлять их под версионный контроль (**git add**), делать коммит изменений (**git commit -m**), отправлять изменения на сервер (**git push**), для получения новых версий файлов репозитория необходимо использовать команду «**git pull**».

17. GitHub является не единственным сервисом, работающим с Git. Какие сервисы ещё Вам известны? Приведите сравнительный анализ одного из таких сервисов с GitHub. GitLab, SourceForge, BitBucket, Launchpad, Apache Allura, Cloud Source, AWS code commit, FogCreek/DevHub, BeanStalk, GitKraken.

Git - это программа контроля версий, а GitLab, GitHub, BitBucket и другие - хостинг просмотра контроля версий, которые нацелены не на локальный контроль версий, а на онлайн, с возможностью коллективной работы.

18. Интерфейс командной строки является не единственным и далеко не самым удобным способом работы с Git. Какие Вам известны программные средства с графическим интерфейсом пользователя для работы с Git? Приведите как реализуются описанные в лабораторной работе операции Git с помощью одного из таких программных средств. GitHub Desktop, GitKraken, SmartGit, SourceTree.

GitHub Desktop: Рабочий стол GitHub в основном является расширением рабочего процесса GitHub. Можно входить в систему, используя свои учётные данные GitHub и начинать работу с репозиториями. Есть возможность создавать новые репозитории, добавлять локальные репозитории и выполнять большинство операций Git из пользовательского интерфейса. GitHub Desktop является полностью открытым исходным кодом, и он доступен для MacOS и Windows.