

Министерство науки и высшего образования Российской Федерации  
Федеральное государственное автономное образовательное учреждение  
высшего образования  
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт цифрового развития  
Кафедра инфокоммуникаций

**ОТЧЕТ**  
**ПО ЛАБОРАТОРНОЙ РАБОТЕ №1**  
**дисциплины «Программирование на Python»**  
**Вариант \_\_\_\_**

Выполнил:  
Репкин Александр Павлович  
1 курс, группа ИВТ-б-о-22-1,  
09.03.01 «Информатика и  
вычислительная техника»,  
направленность (профиль)  
«Программное обеспечение средств  
вычислительной  
техники и автоматизированных  
систем», очная форма обучения

---

(подпись)

Руководитель практики:  
Воронкин Р.А., канд. техн. наук,  
доцент кафедры инфокоммуникаций

---

(подпись)

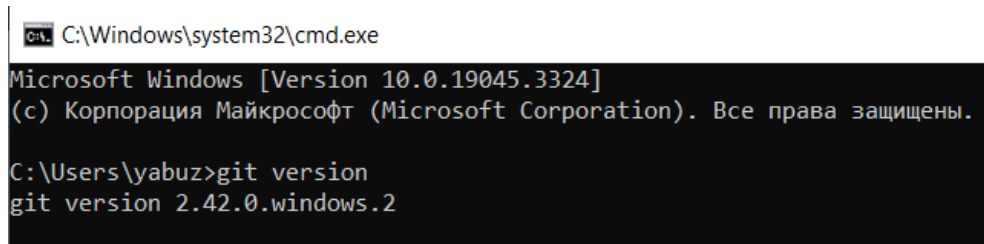
Отчет защищен с оценкой \_\_\_\_\_ Дата защиты \_\_\_\_\_  
Ставрополь, 2023 г.

**Тема:** Исследование основных возможностей Git и GitHub.

**Цель:** исследовать базовые возможности системы контроля версий Git и веб-сервиса для хостинга IT-проектов GitHub.

**Порядок выполнения работы:**

1. Установлен Git. Его работоспособность проверена командой `git version` в командной строке Windows.

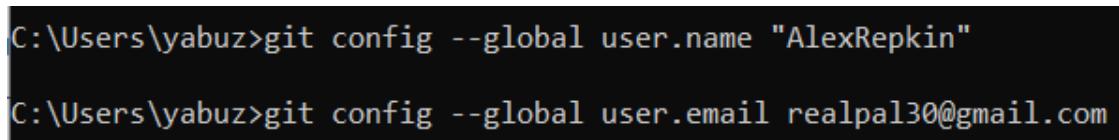


```
C:\Windows\system32\cmd.exe
Microsoft Windows [Version 10.0.19045.3324]
(c) Корпорация Майкрософт (Microsoft Corporation). Все права защищены.

C:\Users\yabuz>git version
git version 2.42.0.windows.2
```

Рисунок 1. Команда `git version`.

2. После проверки, так же с помощью командной строки были изменены имя и электронная почта учётной записи GitHub.



```
C:\Users\yabuz>git config --global user.name "AlexRepkin"
C:\Users\yabuz>git config --global user.email realpal30@gmail.com
```

Рисунок 2. Имя и почта учётной записи.

3. Создан первый репозиторий. Произведено его клонирование на компьютер.

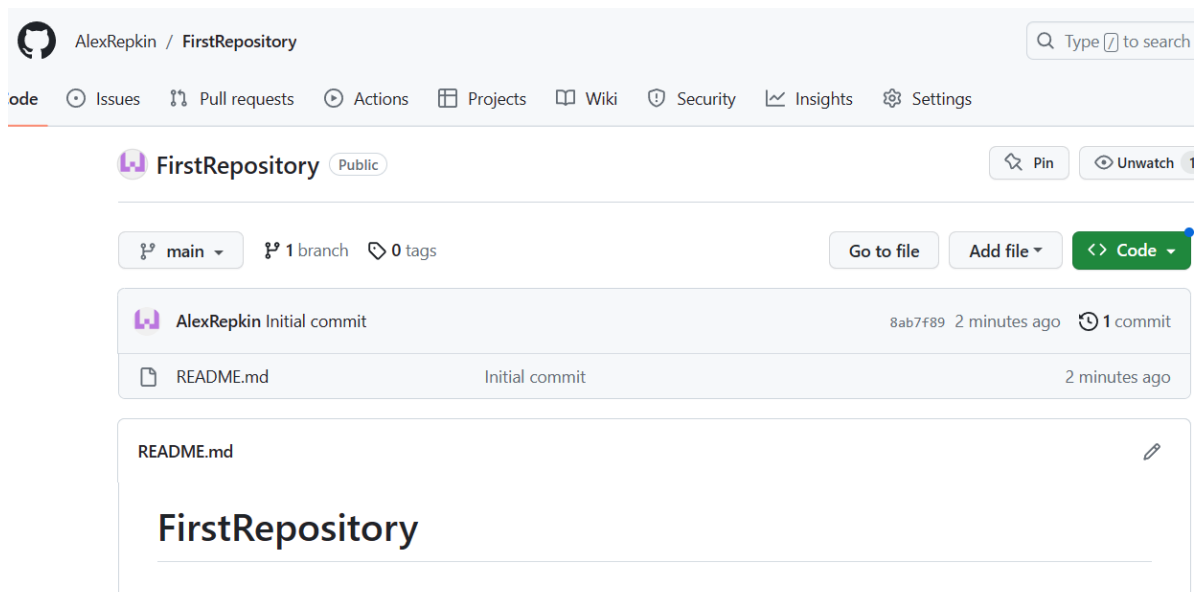


Рисунок 3. Новый репозиторий.

```
C:\Users\yabuz>git clone https://github.com/AlexRepkin/FirstRepository.git
Cloning into 'FirstRepository'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (3/3), done.
```

Рисунок 4. Клонирование файлов на ПК.

4. Получен статус копии репозитория. Для этого было необходимо перейти в его директорию, используя команду `cd` “Название репозитория”.

```
C:\Users\yabuz>cd FirstRepository

C:\Users\yabuz\FirstRepository>git status
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean
```

Рисунок 5. Статус репозитория.

5. Внесены изменения в файл `readme.md` и проверен его статус с помощью командной строки. Используя команду `add`, были внесены изменения, а с помощью команды `commit` – зафиксированы внесённые изменения.

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

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   README.md

no changes added to commit (use "git add" and/or "git commit -a")
```

Рисунок 6. Внесены изменения в README файл.

```
C:\Users\yabuz\FirstRepository>git add README.md

C:\Users\yabuz\FirstRepository>git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   README.md
```

Рисунок 7. Изменения добавлены.

```
C:\Users\yabuz\FirstRepository>git commit -m "Added information about local repository in README file"
[main 0f39fcc] Added information about local repository in README file
1 file changed, 2 insertions(+), 1 deletion(-)
```

Рисунок 8. Использована команда commit.

6. С помощью команды push, после аутентификации, были отправлены изменения с локального репозитория на исходный.

```
C:\Users\yabuz\FirstRepository>git push
info: please complete authentication in your browser...
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Writing objects: 100% (3/3), 316 bytes | 158.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/AlexRepkin/FirstRepository.git
8ab7f89..0f39fcc main -> main
```

Рисунок 9. Использована команда push.

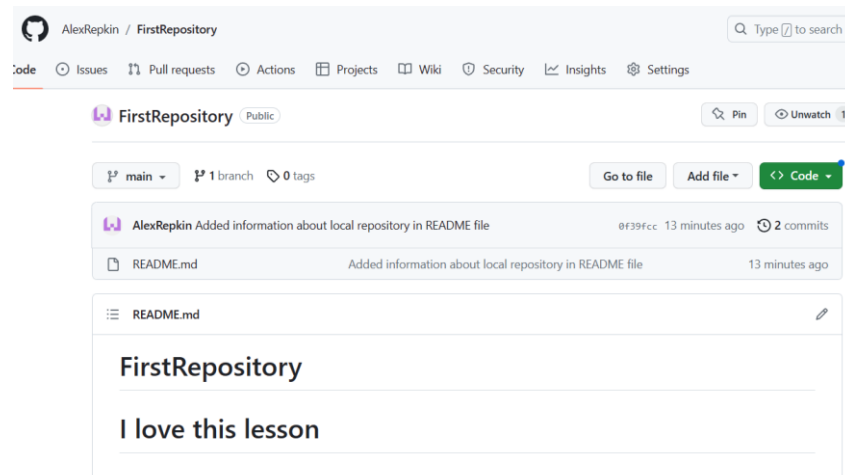


Рисунок 10. Внесённые изменения в исходный репозиторий.

7. Создан новый репозиторий с лицензией MIT. Внесены изменения в файл readme.md – добавлены ФИО и название группы.

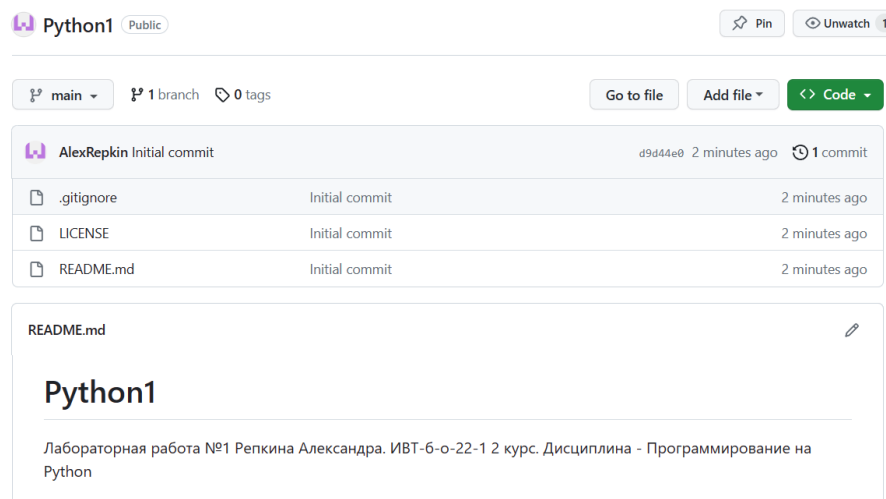


Рисунок 11. Новый репозиторий с лицензией и readme.md.

8. Написана маленькая игра с использованием C++. По мере написания кода использовалась функция `commit` для сохранения достигнутых результатов. Всего было использовано 7 `commit`.

9. Изменения отправлены на сервер с помощью команды `git push`.

10. Ответы на вопросы.

1) Что такое СКВ и каково ее назначение?

**Ответ:** СКВ – система контроля версий, 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?

**Ответ:** локальная система сильно подвержена появлению ошибок. К примеру, можно случайно заменить, скопировать или удалить не тот файл. Централизованная система опасна её возможным отключением на неопределённый срок, из-за чего не будет доступа к файлам у всех участников.

3) К какой СКВ относится Git?

**Ответ:** GIT относится к РСКВ – распределённой системе контроля версий, представляющей собой объединение центральной и локальной СКВ.

4) В чем концептуальное отличие Git от других СКВ?

**Ответ:** главное отличие – подход к работе с данными. Концептуально, большинство других систем хранят информацию в виде списка изменений в файлах. Эти системы представляют хранимую информацию в виде набора файлов и изменений, сделанных в каждом файле, по времени (это контроль версий, основанный на различиях). В Git, вместо этого, каждый раз, когда разработчик делает `commit`, система запоминает, как выглядит каждый файл в этот момент, и сохраняет ссылку на этот снимок. Для увеличения эффективности, если файлы не были изменены, Git не запоминает эти файлы вновь, а только создаёт ссылку на предыдущую версию идентичного файла, который уже сохранён.

5) Как обеспечивается целостность хранимых данных в Git?

**Ответ:** 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. Пользователь не потеряет информацию во время её передачи и не получит повреждённый файл без ведома Git.

**6)** В каких состояниях могут находиться файлы в Git? Как связаны эти состояния?

**Ответ:** доступно три возможных состояния файлов - зафиксированное (committed – файл сохранён в локальной системе), изменённое (modified – файл изменён, но не сохранён в системе), подготовленное (staged – изменённые файлы, которые будут обновлены при следующей фиксации).

**7)** Что такое профиль пользователя в GitHub?

**Ответ:** профиль пользователя – публичная страница человека на GitHub, на которой хранятся его данные, файлы, информация, и к которой можно легко и быстро получить доступ.

**8)** Какие бывают репозитории в GitHub?

**Ответ:** репозитории могут быть локальными, централизованными, распределёнными. Также, репозиторий можно сделать приватным или публичным.

**9)** Укажите основные этапы модели работы с GitHub.

**Ответ:** регистрация – создание и настройка репозитория – клонирование репозитория и его модификация.

**10)** Как осуществляется первоначальная настройка Git после установки?

**Ответ:** после установки Git нужно добавить в его настройки адрес электронной почты и имя пользователя. Сделать это можно при помощи команд:

```
git config --global user.name "Имя пользователя"
```

`git config --global user.email` адрес почты

**11)** Опишите этапы создания репозитория в GitHub.

**Ответ:** после создания аккаунта на GitHub необходимо нажать на New repository. На открывшейся странице необходимо выбрать название репозитория, уникальное для проектов пользователя. Также, можно выбрать Лицензию и .gitignore. При желании можно добавить описание проекту и сделать его приватным. По завершении настраивания, можно нажать на Create Repository.

**12)** Какие типы лицензий поддерживаются GitHub при создании репозитория?

**Ответ:**

1. Apache License 2.0
2. GNU General Public License v3.0
3. MIT License
4. BSD 2-Clause "Simplified" License
5. BSD 3-Clause "New" or "Revised" License
6. Boost Software License 1.0
7. Creative Commons Zero v1.0 Universal
8. Eclipse Public License 2.0
9. GNU Affero General Public License v3.0
10. GNU General Public License v2.0
11. GNU Lesser General Public License v2.1
12. Mozilla Public License 2.0
13. The Unlicense

**13)** Как осуществляется клонирование репозитория GitHub? Зачем нужно клонировать репозиторий?

**Ответ:** клонирование можно произвести в командной строке с помощью команды `git clone "адрес"`. Клонирование создаёт локальную копию проекта, что позволяет работать с ней, не изменяя оригинал и не мешая остальным участникам разработки.

**14) Как проверить состояние локального репозитория Git?**

**Ответ:** для проверки состояния локального репозитория используется команда `git status`. Важно помнить, что выполняться она должна в папке репозитория.

**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`?**

**Ответ:** после добавления/изменения файла в локальный репозиторий Git, Git начинает следить за этим файлом. Он будет отображаться как измененный файл при выполнении команды `git status`.

После добавления нового/измененного файла под версионный контроль с помощью команды `git add`, файл добавляется в индекс (промежуточное хранилище перед фиксацией изменений). Файл будет готов к фиксации (`commit`) и будет отображаться как измененный файл при выполнении команды `git status`.

После фиксации (`commit`), все файлы, добавленные в индекс с помощью `git add`, становятся частью истории репозитория. Этот коммит будет отображаться при выполнении команды `git log`.

После отправки изменений на сервер с помощью команды `git push`, состояние локального репозитория Git синхронизируется с удаленным репозиторием. Все коммиты, которых нет на удаленном репозитории, будут отправлены на сервер, и удаленный репозиторий будет обновлен с новыми изменениями.

**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`.

**Ответ:** на обоих компьютерах необходимо создать локальную копию репозитория. Сделать это можно с помощью `git clone` “ссылка”. Когда один из компьютеров использует команду `git push`, второе устройство может использовать команду `git pull` для получения более свежей версии репозитория.

**17)** GitHub является не единственным сервисом, работающим с Git. Какие сервисы еще Вам известны? Приведите сравнительный анализ одного из таких сервисов с GitHub.

**Ответ:** существуют также сервисы – GitLab, BitBucket. Согласно данным и отзывам, у BitBucket присутствует возможность редактирования и написания кода прямо в онлайн-редакторе. Также, BitBucket более ориентирован на малые проекты, в то время как крупные команды предпочитают GitHub.

**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.

**Ответ:**

GitHub Desktop - графический клиент Git, разработанный специально для работы с GitHub. Для выполнения операции создания новой ветки в GitHub Desktop необходимо выбрать репозиторий, перейти во вкладку "Branch", нажать на кнопку "New branch" и ввести имя новой ветки. Затем GitHub Desktop создает новую ветку как на компьютере, так и на GitHub.

GitKraken - графический интерфейс для выполнения всех основных операций Git. Для выполнения операции клонирования репозитория в GitKraken необходимо нажать на кнопку "Open a repository", выбрать "Clone",

указать URL репозитория и директорию назначения. Затем GitKraken клонирует репозиторий на компьютер.

**Вывод:** в ходе выполнения лабораторной работы были исследованы базовые возможности системы контроля версий Git и веб-сервиса для хостинга IT-проектов GitHub.