

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ
ФЕДЕРАЦИИ**

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ**

ВЫСШЕГО ОБРАЗОВАНИЯ

**«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ» ИНСТИТУТ
ЦИФРОВОГО РАЗВИТИЯ**

**Отчет о лабораторной работе №1.1 по дисциплине основы программной
инженерии**

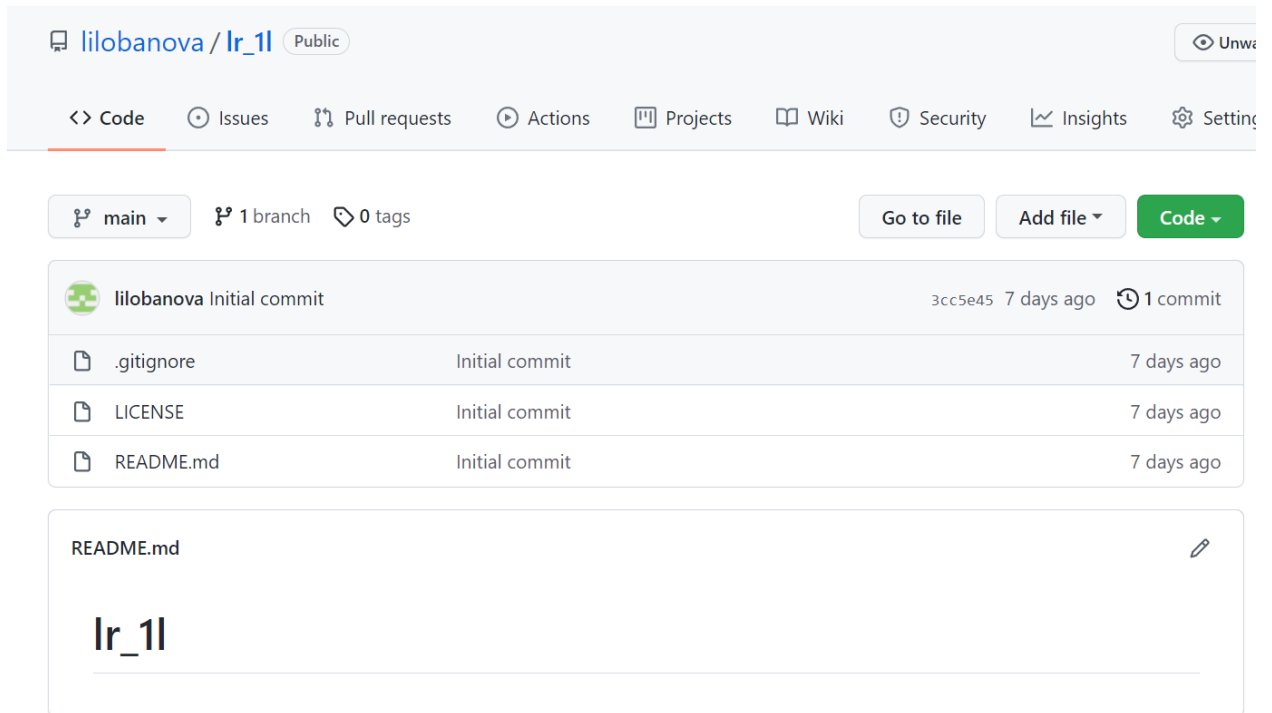
Выполнила:
Лобанова Елизавета
Евгеньевна,
2 курс, группа ПИЖ-б-о-20-1

Проверил:
Доцент кафедры прикладной
математики и
компьютерной безопасности,
Воронкин Р.А.

Отчет защищен с оценкой _____ Дата защиты _____

Ставрополь, 2021 г.

Ход работы



Создание репозитория

```
C:\Users\Acer\OneDrive\Рабочий стол\lr_1_opi> cd lr_1l
C:\Users\Acer\OneDrive\Рабочий стол\lr_1_opi\lr_1l>git clone C:\Users\Acer\OneDrive\Рабочий стол\lr_1_opi
Cloning into 'стол/lr_1_opi'...
fatal: 'C:\Users\Acer\OneDrive\Рабочий' does not appear to be a git repository
fatal: could not read from remote repository.
```

Копирование репозитория на компьютер

```
C:\Users\Acer\OneDrive\Рабочий стол\lr_1_opi\lr_1l>git status
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean
C:\Users\Acer\OneDrive\Рабочий стол\lr_1_opi\lr_1l>
```

Команда status

```
C:\Users\Acer\OneDrive\Рабочий стол\lr_1_opi\lr_1l>git add .
warning: LF will be replaced by CRLF in стол/opil/opil.cbp.
The file will have its original line endings in your working directory
```

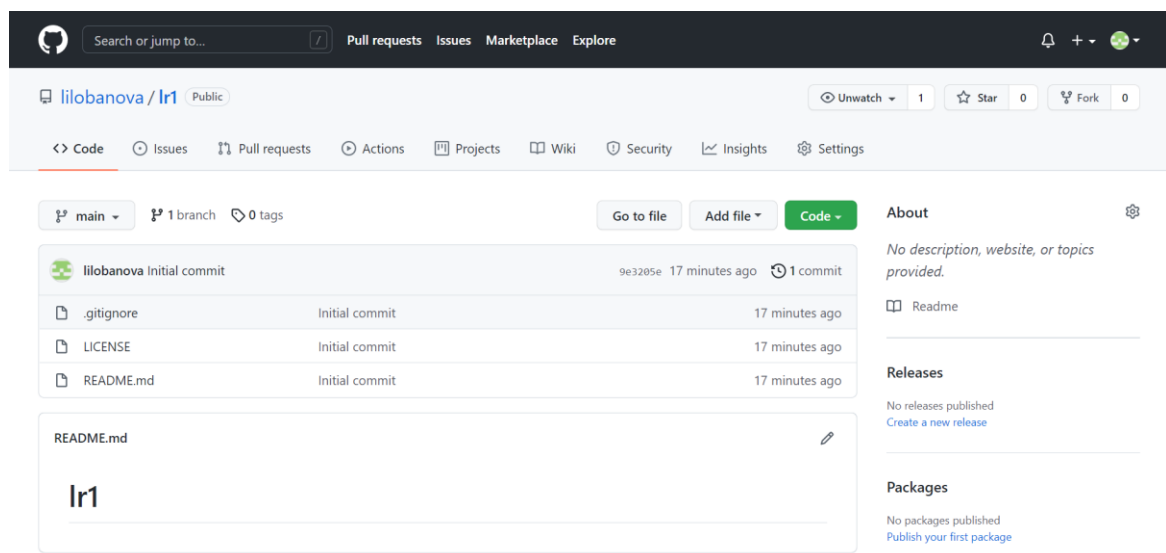
Команда add

```
C:\Users\Acer\OneDrive\Рабочий стол\lr_1_opi\lr_1l>git commit -m "Команда push"
[main 62a9633] Команда push
3 files changed, 82 insertions(+)
create mode 100644 main.cpp
create mode 100644 "\321\201\321\202\320\276\320\273\opi1/main.cpp"
create mode 100644 "\321\201\321\202\320\276\320\273\opi1\opi1.cbp"
```

Команда commit

```
C:\Users\Acer\OneDrive\Рабочий стол\lr_1_opi\lr_1l>git push origin main
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 8 threads
Compressing objects: 100% (5/5), done.
Writing objects: 100% (6/6), 1.02 KiB | 521.00 KiB/s, done.
Total 6 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/lilobanova/lr_1l.git
3cc5e45..62a9633 main -> main
```

Команда push



The screenshot shows the GitHub interface for a repository named 'lilobanova/lr1'. The repository is public and has 1 branch (main) and 0 tags. The initial commit, 9e3205e, was made 17 minutes ago. The commit message is 'Initial commit'. The files included in the commit are .gitignore, LICENSE, and README.md. The README.md file is visible, showing the text 'lr1'. The right sidebar contains sections for 'About' (No description, website, or topics provided), 'Releases' (No releases published), and 'Packages' (No packages published).

Первоначальный репозиторий

Search or jump to...

7

Pull requests

Issues

Marketplace

Explore

lilobanova / lr1

Public

Unwatch1Star0Fork0

<> Code

Issues

Pull requests

Actions

Projects

Wiki

Security

Insights

Settings

main1 branch0 tags

Go to fileAdd fileCode

lilobanova

Добавила заголовок

4be92c33 minutes ago2 commits

.gitignore	Initial commit	18 minutes ago
LICENSE	Initial commit	18 minutes ago
README.md	Добавила заголовок	3 minutes ago

README.md

lr1

##Заголовок

About

No description, website, or topics provided.

ReadmeMIT License

Releases

No releases published
[Create a new release](#)

Packages

No packages published

Измененный репозиторий

lilobanova

Merge branch 'main' of https://github.com/lilobanova/lr_1l

4ce18d41 minute ago7 commits

стол/opi1	Команда push	15 minutes ago
.gitignore	Initial commit	7 days ago
LICENSE	Initial commit	7 days ago
README.md	Rename Лр1 to README.md	6 minutes ago
main.cpp	Команда pop	7 minutes ago

Добавление нового commit на сервере

```

C:\Users\Acer\OneDrive\Рабочий стол\lr_1_opi\lr_1l>git commit -m "Команды back и size"
[main a4984df] Команды back и size
1 file changed, 8 insertions(+)

C:\Users\Acer\OneDrive\Рабочий стол\lr_1_opi\lr_1l>git push
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 8 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 377 bytes | 377.00 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/lilobanova/lr_1l.git
4ce18d4..a4984df main -> main

C:\Users\Acer\OneDrive\Рабочий стол\lr_1_opi\lr_1l>git add .

C:\Users\Acer\OneDrive\Рабочий стол\lr_1_opi\lr_1l>git commit -m "Команда empty"
[main 28f16c7] Команда empty
1 file changed, 6 insertions(+)

C:\Users\Acer\OneDrive\Рабочий стол\lr_1_opi\lr_1l>git push
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 8 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 357 bytes | 357.00 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/lilobanova/lr_1l.git
a4984df..28f16c7 main -> main

C:\Users\Acer\OneDrive\Рабочий стол\lr_1_opi\lr_1l>git add .

C:\Users\Acer\OneDrive\Рабочий стол\lr_1_opi\lr_1l>gitcommit -m "Команда bye"
"gitcommit" не является внутренней или внешней
командой, исполняемой программой или пакетным файлом.

C:\Users\Acer\OneDrive\Рабочий стол\lr_1_opi\lr_1l>git commit -m "Команда bye"
[main 479b7f5] Команда bye
1 file changed, 5 insertions(+), 1 deletion(-)

C:\Users\Acer\OneDrive\Рабочий стол\lr_1_opi\lr_1l>git push
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 8 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 340 bytes | 340.00 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/lilobanova/lr_1l.git
28f16c7..479b7f5 main -> main

```

Добавление новых commits

main
1 branch
0 tags
Go to file
Add file
Code

lilobanova Команда bye
479b7f5 2 minutes ago
10 commits

стол/opi1	Команда push	2 hours ago
.gitignore	Initial commit	7 days ago
LICENSE	Initial commit	7 days ago
README.md	Rename Лр1 to README.md	2 hours ago
main.cpp	Команда bye	2 minutes ago

README.md

Выполнила Лобанова Елизавета Группы ПИЖ-6-о-20-1

Результат добавления 10 commits

Ответы на вопросы

1. Что такое СКВ и каково ее 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. Программисты обычно помещают в систему контроля версий исходные коды программ, но на самом деле под версионный контроль можно поместить файлы практически любого типа.

2. В чем недостатки локальных и централизованных СКВ?

Администраторы имеют полный контроль над тем, кто и что может делать, и гораздо проще администрировать ЦСКВ, чем оперировать локальными базами данных на каждом клиенте. Несмотря на это, данный подход тоже имеет серьёзные минусы. Самый очевидный минус — 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, над которыми работает, а также никто не сможет обмениваться этими изменениями с другими разработчиками. Если жёсткий диск, на котором хранится центральная БД, повреждён, а своевременные бэкапы отсутствуют, вы потеряете всё — всю историю проекта, не считая единичных снимков репозитория, которые сохранились на локальных машинах разработчиков. Локальные СКВ страдают от той же самой проблемы: когда вся история проекта хранится в одном месте, вы рискуете потерять всё.

3. К какой СКВ относится Git?

Распределённые системы контроля версий (РСКВ). В РСКВ (таких как Git, Mercurial, Bazaar или Darcs)

4. В чем концептуальное отличие Git от других СКВ?

Основное отличие Git от любой другой СКВ (включая Subversion и её собратьев) — это подход к работе со своими данными. Концептуально, большинство других систем хранят информацию в виде списка изменений в файлах. Эти системы (CVS, Subversion, Perforce, Bazaar и т. д.) представляют хранимую информацию в виде набора файлов и изменений, сделанных в каждом файле, по времени (обычно это называют контролем версий, основанным на различиях)

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. Вы не потеряете информацию во время её передачи и не получите повреждённый файл без ведома Git.

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

У Git есть три основных состояния, в которых могут находиться ваши 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.

7. Что такое профиль пользователя в GitHub?

Профиль - это ваша публичная страница на GitHub, как и в социальных сетях. Когда вы ищете работу в качестве программиста, работодатели могут посмотреть ваш профиль GitHub и принять его во внимание, когда будут решать, брать вас на работу или нет.

8. Какие бывают репозитории в GitHub?

Публичный и приватный

9. Укажите основные этапы модели работы с GitHub.

Две отдельные области: GitHub и ваш компьютер. Сначала рассмотрим область GitHub. В нем есть два хранилища: upstream - это оригинальный репозиторий проекта, который вы скопировали, origin - ваш fork (копия) на GitHub, к которому у вас есть полный доступ.

Чтобы перенести изменения с вашей копии в исходному репозиторий проекта, вам нужно сделать запрос на извлечение. Если вы хотите внести небольшие изменения в свою копию (fork), вы можете использовать вебинтерфейс GitHub. Однако такой

подход не удобен при разработке программ, поскольку вам часто приходится запускать и отлаживать их локально. Стандартный способ - создать локальный клон удаленного репозитория и работать с ним локально, периодически внося изменения в удаленный репозиторий.

10. Как осуществляется первоначальная настройка Git после установки? После создания репозитория его необходимо клонировать на ваш компьютер.

git version. После этого ввести следующие команды:

```
git config --global user.name <YOUR_NAME>
```

```
git config --global user.email <EMAIL>
```

11. Опишите этапы создания репозитория в GitHub.

В результате будет выполнен переход на страницу создания репозитория. Наиболее важными на ней являются следующие поля: Имя репозитория. Оно может быть любое, необязательно уникальное во всем github, потому что привязано к вашему аккаунту, но уникальное в рамках тех репозиторий, которые вы создавали. Описание (Description). Можно оставить пустым. Public/private. Выбираем открытый (Public), НЕ ставим галочку "Initialize this repository with a README" (В README потом будет лежать какая-то основная информация, что же такое ваш проект и как с ним работать). .gitignore и LICENSE можно сейчас не выбирать.

После заполнения этих полей нажимаем кнопку Create repository. Отлично, ваш репозиторий готов!

12. Какие типы лицензий поддерживаются GitHub при создании репозитория?

gitignore и LICENSE

13. Как осуществляется клонирование репозитория GitHub? Зачем нужно клонировать репозиторий?

Кнопка Clone или Code, нужно щелкнуть по ней и скопировать ссылку, затем в GIT.cmd через команду git clone вставить ссылку и скопировать репозиторий на ПК.

14. Как проверить состояние локального репозитория Git?

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 ?

После изменения в файле, при вводе команды git status данный файл будет подсвечен красным цветом, говоря о том, что изменения в нем не отображены в локальном репозитории. После выполнения команды git add название файла будет подсвечено зеленым, а текст в консоли будет говорить о том, что есть изменения нуждающиеся в комите. После выполнения команды git commit будет отображено, что текущая

ветвь находится на определенное количество коммитов дальше исходной. После отправки изменений будет отображено, что “конец” удаленной ветки находится там, же где и результат последних коммитов.

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`.

`git clone` написать на обоих компьютерах.

При изменении файла на одном компьютере нужно зафиксировать сохранение на сервере через `git commit` и `git push`.

Для изменения этого же файла на втором компьютере нужно написать `git pull`.

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

GitLab, Bitbucket, Beanstalk. К примеру, пользователь GitLab может создать сколько угодно частных репозиториев. Основное различие между GitHub и GitLab заключается в философии, которую представляет каждая платформа. GitHub имеет более высокую доступность и в большей степени ориентирован на производительность инфраструктуры, в то время как GitLab в большей степени ориентирован на предложение системы, основанной на функциях, с централизованной, интегрированной платформой для веб-разработчиков.

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.

Git поставляется со встроенными инструментами с графическим интерфейсом для фиксации (`git-gui`) и просмотра (`gitk`), но есть несколько сторонних инструментов для пользователей, которым нужен опыт работы с конкретной платформой.

GitKraken - популярный клиент с графическим интерфейсом Git для Windows, Mac и Linux. Это бесплатно для некоммерческого использования. Это отличный инструмент для новичков и опытных пользователей Git, позволяющий повысить эффективность за счет интуитивно понятного интерфейса, бесшовной интеграции и более быстрого и гибкого рабочего процесса.