

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 №1.1
дисциплины «Основы кроссплатформенного программирования»

Выполнил:
Кононов Дмитрий Александрович
1 курс, группа ИТС-б-о-22-1,
11.03.02 «Инфокоммуникационные
технологии и системы связи»,
направленность (профиль)
«Инфокоммуникационные системы и
сети», очная форма обучения

(подпись)

Руководитель практики:
Воронкин Р. А., 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

Ход работы:

Вариант №17

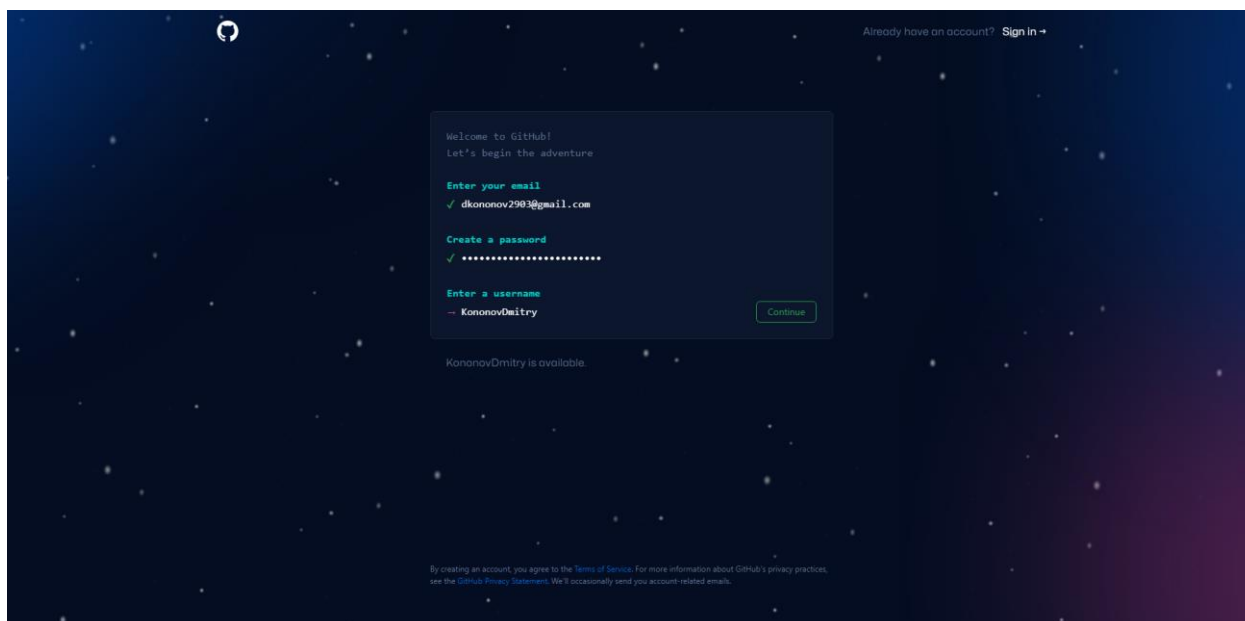


Рисунок 1. Создание аккаунта на GitHub

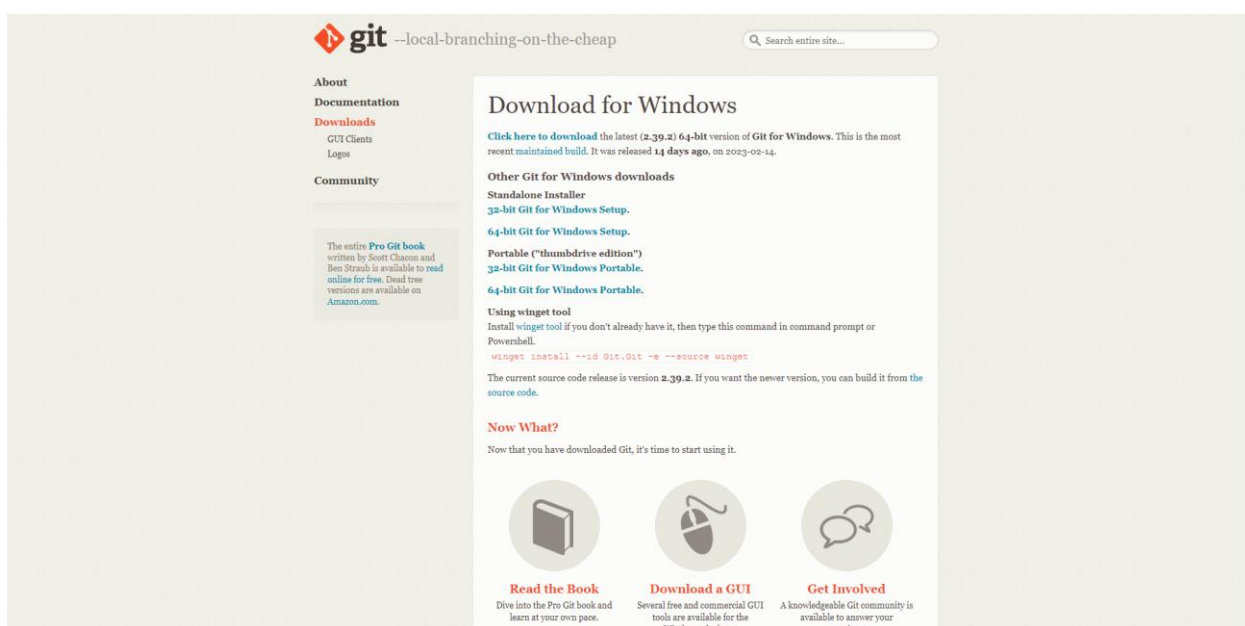
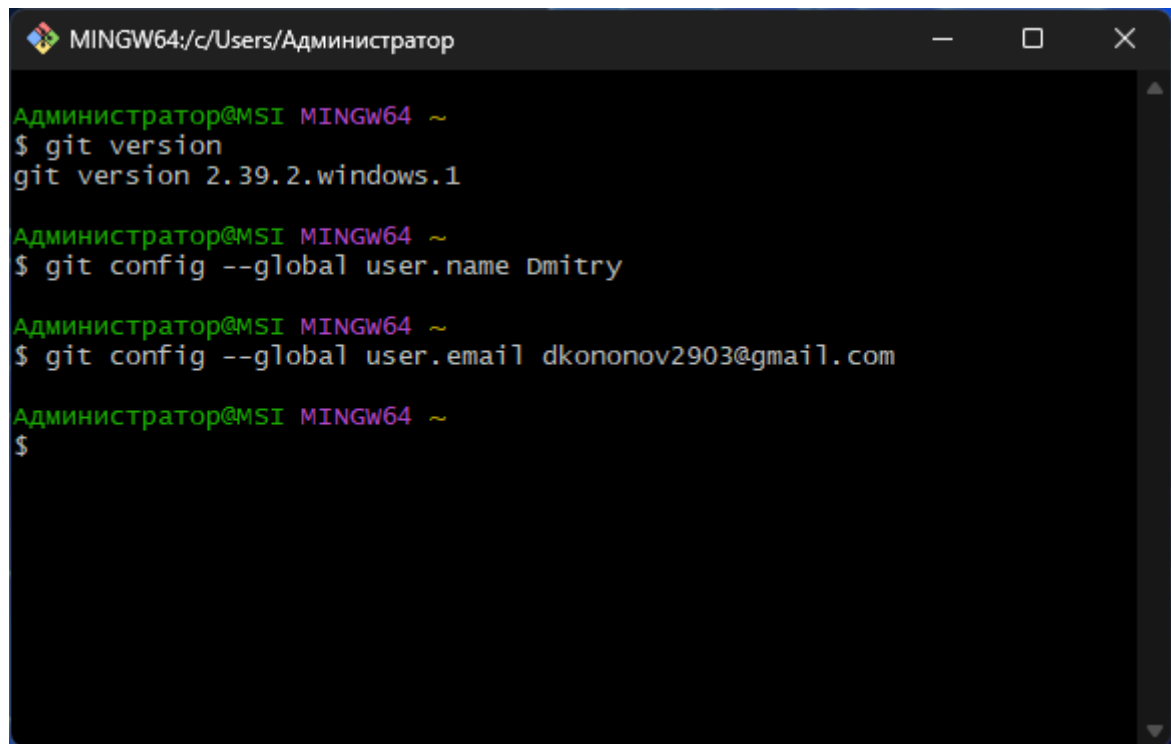


Рисунок 2. Установка Git



```
MINGW64:/c/Users/Администратор

Администратор@MSI MINGW64 ~
$ git version
git version 2.39.2.windows.1

Администратор@MSI MINGW64 ~
$ git config --global user.name Dmitry

Администратор@MSI MINGW64 ~
$ git config --global user.email dkononov2903@gmail.com

Администратор@MSI MINGW64 ~
$
```

Рисунок 3. Текущая версия Git + добавление имени и электронной почты

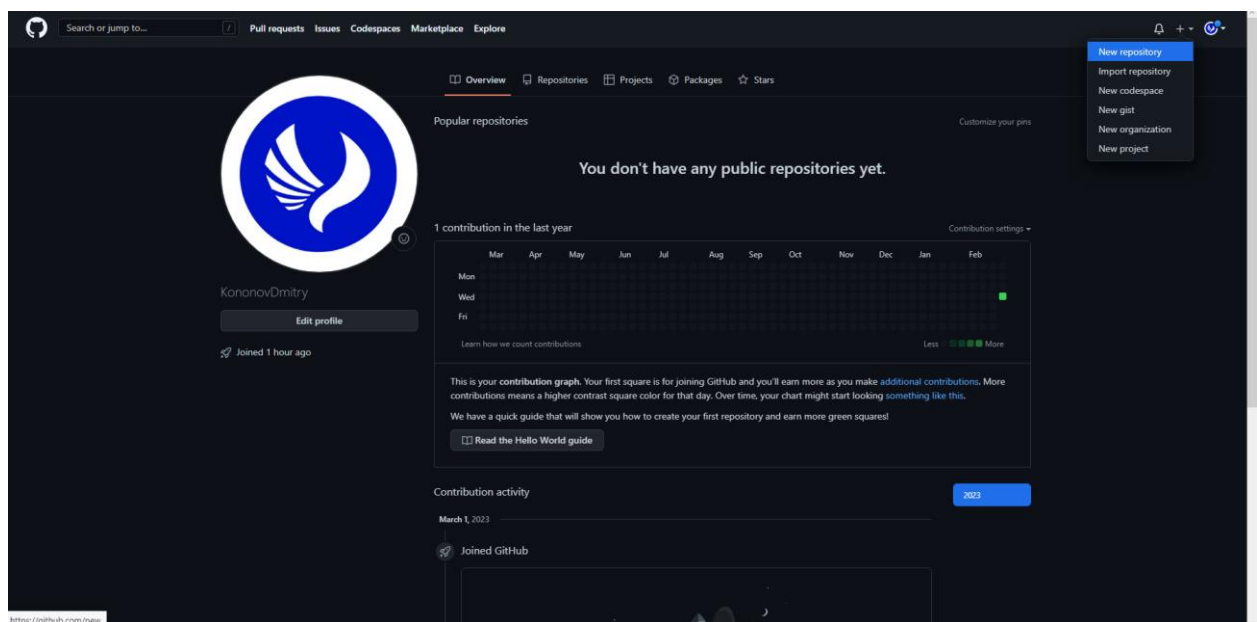


Рисунок 4. Начало создания репозитория

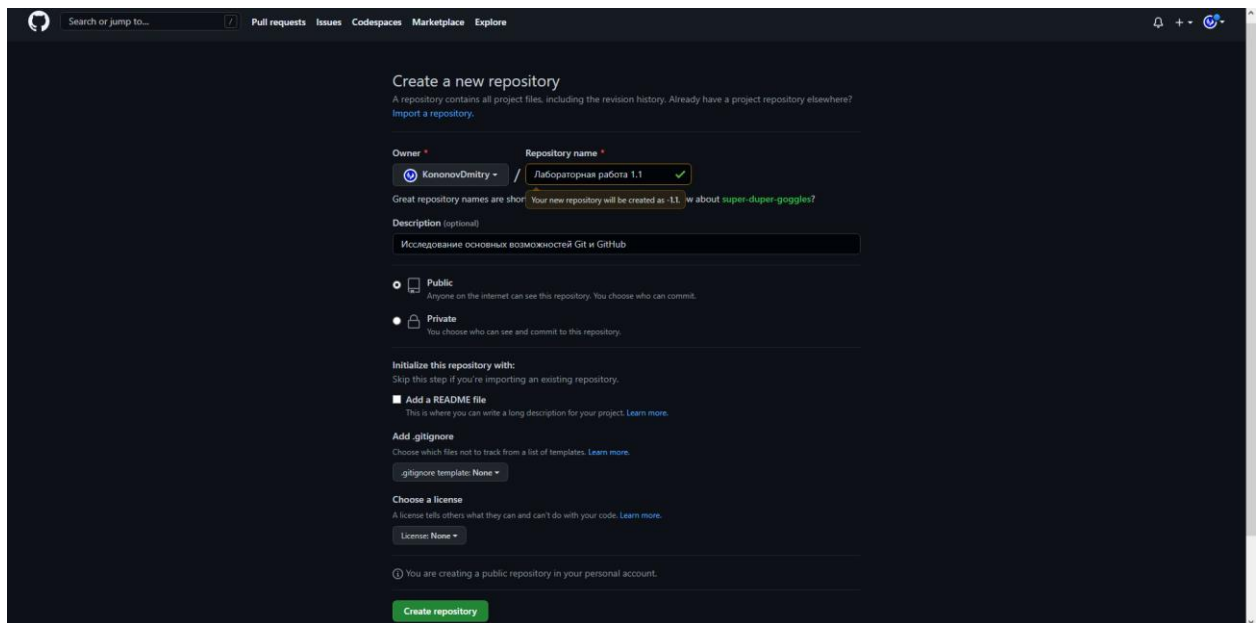


Рисунок 5. Создание репозитория GitHub

```
MINGW64:/c/Users/Администратор

Администратор@MSI MINGW64 ~
$ git version
git version 2.39.2.windows.1

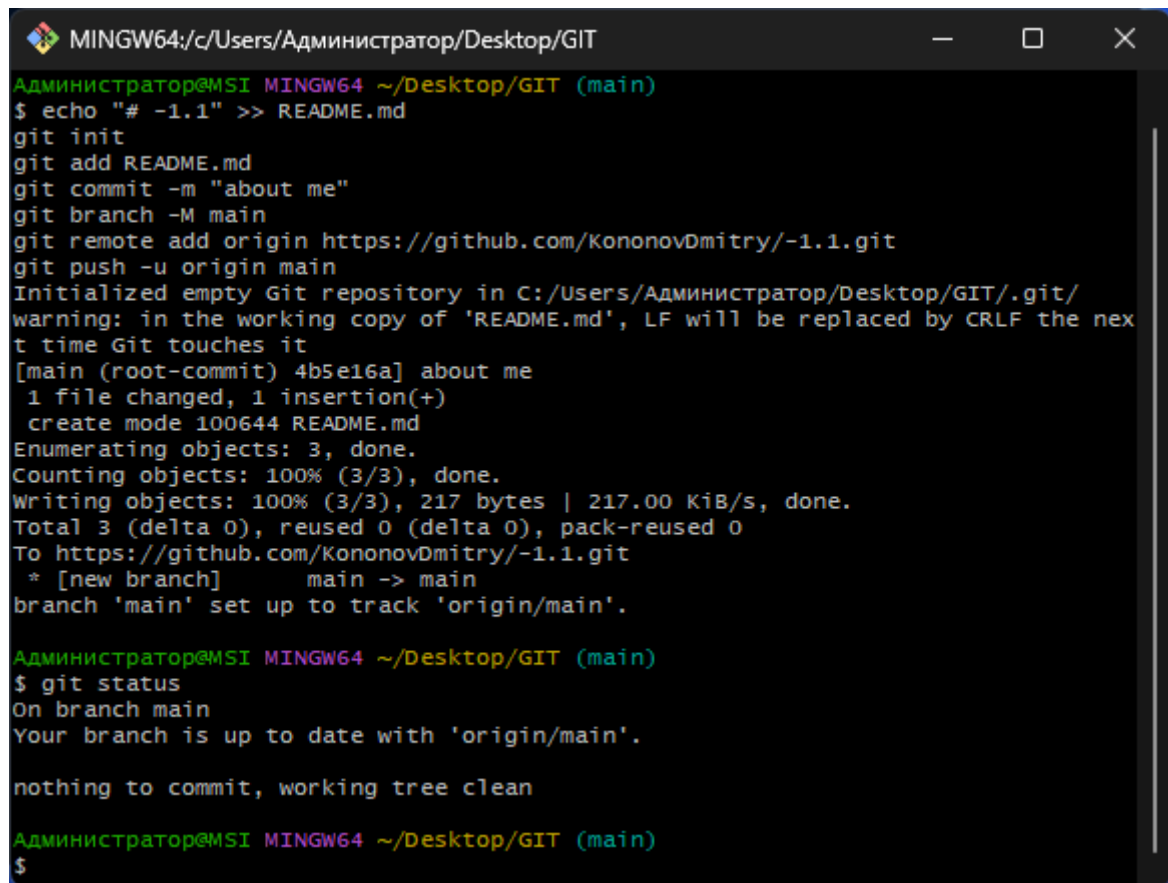
Администратор@MSI MINGW64 ~
$ git config --global user.name Dmitry

Администратор@MSI MINGW64 ~
$ git config --global user.email dkononov2903@gmail.com

Администратор@MSI MINGW64 ~
$ git clone https://github.com/KononovDmitry/-1.1.git
Cloning into '-1.1'...
warning: You appear to have cloned an empty repository.

Администратор@MSI MINGW64 ~
$
```

Рисунок 6. Клонирование репозитория



```
MINGW64:/c/Users/Администратор/Desktop/GIT
Администратор@MSI MINGW64 ~/Desktop/GIT (main)
$ echo "# -1.1" >> README.md
git init
git add README.md
git commit -m "about me"
git branch -M main
git remote add origin https://github.com/KononovDmitry/-1.1.git
git push -u origin main
Initialized empty Git repository in C:/Users/Администратор/Desktop/GIT/.git/
warning: in the working copy of 'README.md', LF will be replaced by CRLF the next time Git touches it
[main (root-commit) 4b5e16a] about me
1 file changed, 1 insertion(+)
create mode 100644 README.md
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 217 bytes | 217.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/KononovDmitry/-1.1.git
 * [new branch]      main -> main
branch 'main' set up to track 'origin/main'.

Администратор@MSI MINGW64 ~/Desktop/GIT (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean

Администратор@MSI MINGW64 ~/Desktop/GIT (main)
$
```

Рисунок 7. Клонирование репозитория

```
MINGW64/c/Users/Администратор/Desktop/GIT
To https://github.com/KononovDmitry/-1.1.git
* [new branch]      main -> main
branch 'main' set up to track 'origin/main'.

Администратор@MSI MINGW64 ~/Desktop/GIT (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean

Администратор@MSI MINGW64 ~/Desktop/GIT (main)
$ 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")

Администратор@MSI MINGW64 ~/Desktop/GIT (main)
$ git add .
warning: in the working copy of 'README.md', LF will be replaced by CRLF the next time Git touches it

Администратор@MSI MINGW64 ~/Desktop/GIT (main)
$ git add README.md

Администратор@MSI MINGW64 ~/Desktop/GIT (main)
$ git commit -m "about me 1"
[main 18fdbd8] about me 1
 1 file changed, 1 insertion(+), 1 deletion(-)

Администратор@MSI MINGW64 ~/Desktop/GIT (main)
$ git push
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 12 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 288 bytes | 288.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/KononovDmitry/-1.1.git
 4b5e16a..18fdbd8  main -> main
```

Рисунок 8. Изменение README и добавление в репозиторий

```
MINGW64:/c/Users/Администратор/Desktop/GIT
Администратор@MSI MINGW64 ~/Desktop/GIT (main)
$ git add .gitignore
warning: in the working copy of '.gitignore', LF will be replaced by CRLF the next time Git touches it

Администратор@MSI MINGW64 ~/Desktop/GIT (main)
$ git add g.gitignore
warning: in the working copy of 'g.gitignore', LF will be replaced by CRLF the next time Git touches it

Администратор@MSI MINGW64 ~/Desktop/GIT (main)
$ ^C

Администратор@MSI MINGW64 ~/Desktop/GIT (main)
$ git config --global core.autocrlf false

Администратор@MSI MINGW64 ~/Desktop/GIT (main)
$ git add .gitignore

Администратор@MSI MINGW64 ~/Desktop/GIT (main)
$ git commit -m "replace .gitignore"
[main 7c70021] replace .gitignore
2 files changed, 370 insertions(+)
create mode 100644 .gitignore
create mode 100644 g.gitignore

Администратор@MSI MINGW64 ~/Desktop/GIT (main)
$ git push
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 12 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (2/2), 315 bytes | 315.00 KiB/s, done.
Total 2 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/KononovDmitry/-1.1.git
402dd37..7c70021 main -> main

Администратор@MSI MINGW64 ~/Desktop/GIT (main)
$
```

Рисунок 9. Добавление файла .gitignore в репозиторий

```
MINGW64:/c/Users/Администратор/Desktop/GIT
Администратор@MSI MINGW64 ~/Desktop/GIT (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

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

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        .gitignore.txt
        programm.py

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

Администратор@MSI MINGW64 ~/Desktop/GIT (main)
$ git add programm.py

Администратор@MSI MINGW64 ~/Desktop/GIT (main)
$ git commit -m "import library"
[main 290f2f1] import library
1 file changed, 1 insertion(+)
create mode 100644 programm.py

Администратор@MSI MINGW64 ~/Desktop/GIT (main)
$ git push
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 12 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 323 bytes | 161.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/KononovDmitry/-1.1.git
   a645c62..290f2f1  main -> main

Администратор@MSI MINGW64 ~/Desktop/GIT (main)
$
```

Рисунок 10. Добавление файла с программой в репозиторий


```
MINGW64:/c/Users/Администратор/Desktop/GIT
Администратор@MSI MINGW64 ~/Desktop/GIT (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes not staged for commit:
  (use "git add/rm <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        deleted:    gitignore
        modified:   programm.py

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        .gitignore.txt

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

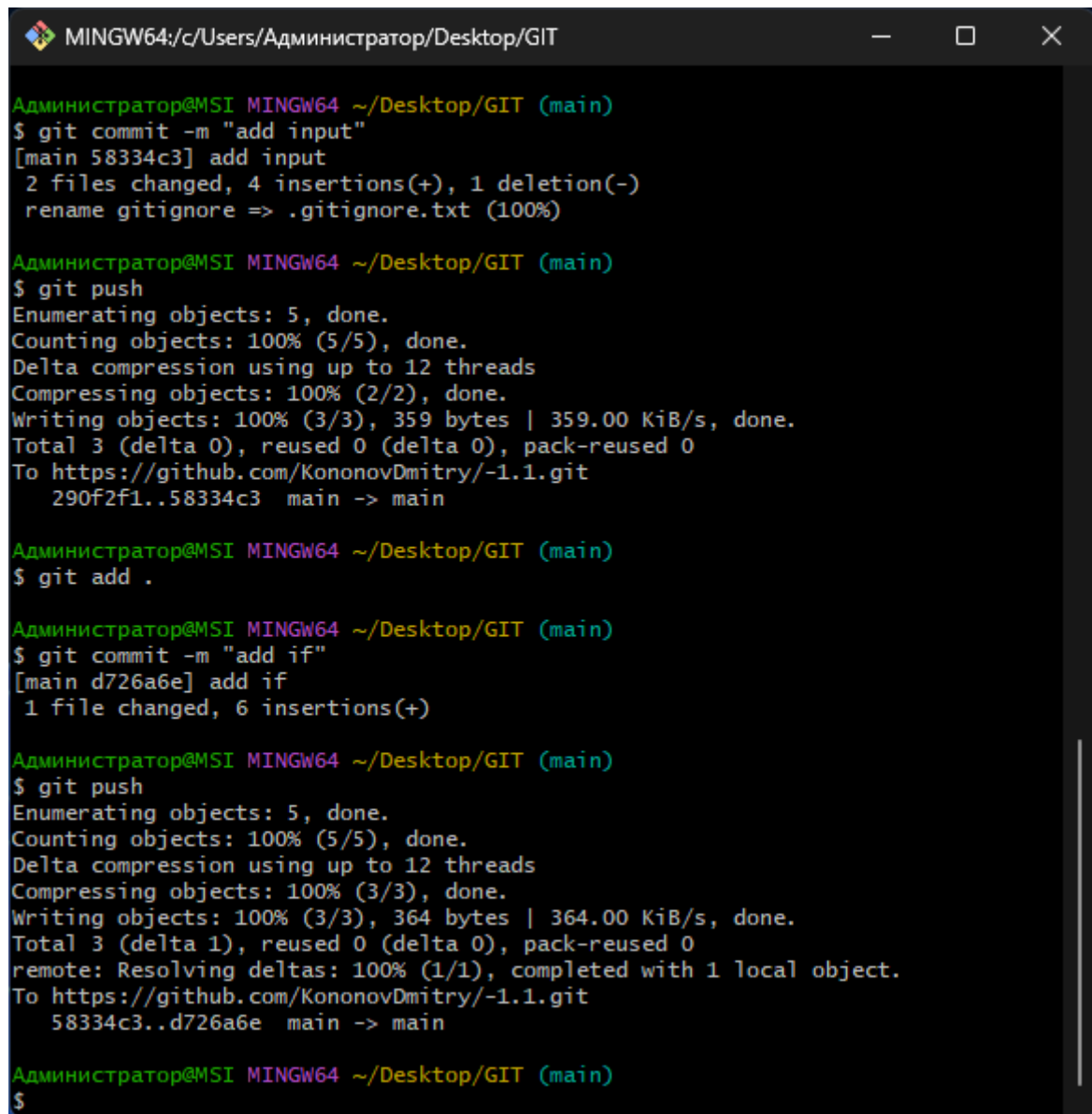
Администратор@MSI MINGW64 ~/Desktop/GIT (main)
$ git add .

Администратор@MSI MINGW64 ~/Desktop/GIT (main)
$ git commit -m "add input"
[main 58334c3] add input
 2 files changed, 4 insertions(+), 1 deletion(-)
 rename gitignore => .gitignore.txt (100%)

Администратор@MSI MINGW64 ~/Desktop/GIT (main)
$ git push
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 12 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 359 bytes | 359.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/KononovDmitry/-1.1.git
   290f2f1..58334c3  main -> main

Администратор@MSI MINGW64 ~/Desktop/GIT (main)
$
```

Рисунок 11. Добавление файла с программой в репозиторий



```
MINGW64:/c/Users/Администратор/Desktop/GIT

Администратор@MSI MINGW64 ~/Desktop/GIT (main)
$ git commit -m "add input"
[main 58334c3] add input
2 files changed, 4 insertions(+), 1 deletion(-)
rename gitignore => .gitignore.txt (100%)

Администратор@MSI MINGW64 ~/Desktop/GIT (main)
$ git push
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 12 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 359 bytes | 359.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/KononovDmitry/-1.1.git
290f2f1..58334c3 main -> main

Администратор@MSI MINGW64 ~/Desktop/GIT (main)
$ git add .

Администратор@MSI MINGW64 ~/Desktop/GIT (main)
$ git commit -m "add if"
[main d726a6e] add if
1 file changed, 6 insertions(+)

Администратор@MSI MINGW64 ~/Desktop/GIT (main)
$ git push
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 12 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 364 bytes | 364.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/KononovDmitry/-1.1.git
58334c3..d726a6e main -> main

Администратор@MSI MINGW64 ~/Desktop/GIT (main)
$
```

Рисунок 12. Добавление файла с программой в репозиторий

```
MINGW64:/c/Users/Администратор/Desktop/GIT
Администратор@MSI MINGW64 ~/Desktop/GIT (main)
$ git commit -m "add if"
[main d726a6e] add if
1 file changed, 6 insertions(+)

Администратор@MSI MINGW64 ~/Desktop/GIT (main)
$ git push
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 12 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 364 bytes | 364.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/KononovDmitry/-1.1.git
58334c3..d726a6e main -> main

Администратор@MSI MINGW64 ~/Desktop/GIT (main)
$ git add .

Администратор@MSI MINGW64 ~/Desktop/GIT (main)
$ git commit -m "add else"
[main 402dd37] add else
1 file changed, 3 insertions(+)

Администратор@MSI MINGW64 ~/Desktop/GIT (main)
$ git push
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 12 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 338 bytes | 338.00 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/KononovDmitry/-1.1.git
d726a6e..402dd37 main -> main

Администратор@MSI MINGW64 ~/Desktop/GIT (main)
$
```

Рисунок 13. Добавление файла с программой в репозиторий

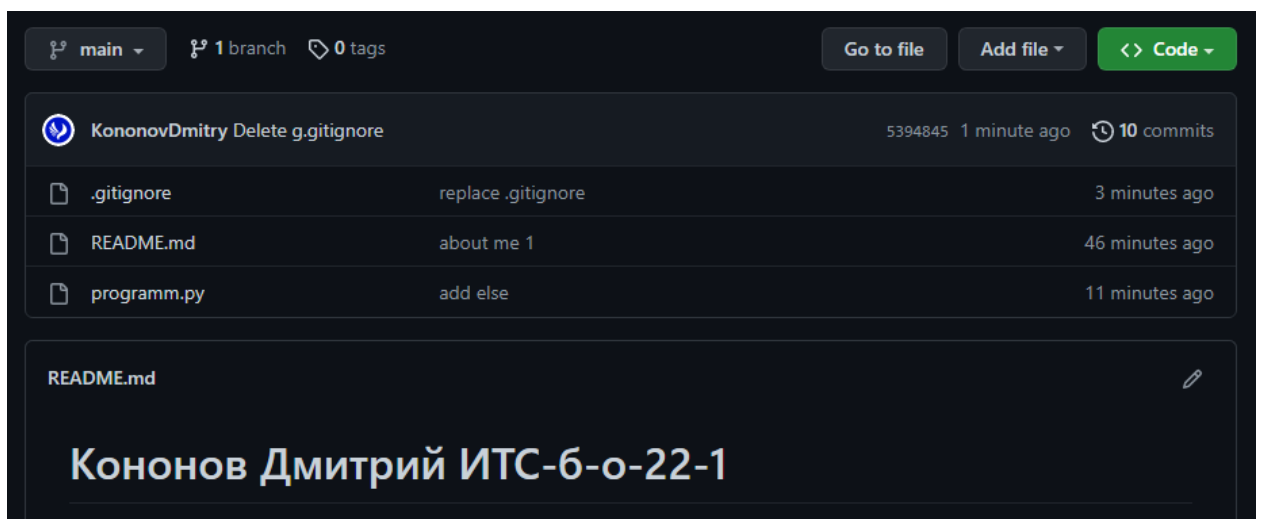


Рисунок 14. Обновленный репозиторий

Ссылка: <https://github.com/KononovDmitry/-1.1>

Ответы на контрольные вопросы:

1. Что такое СКВ и каково ее назначение?

Система контроля версий (СКВ) — это система, регистрирующая изменения в одном или нескольких файлах этих файлов.

2. В чем недостатки локальных и централизованных 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, над которыми работает, а также никто не сможет обмениваться этими изменениями с другими разработчиками.

3. К какой СКВ относится Git?

К распределённым системам контроля версий.

4. В чем концептуальное отличие Git от других СКВ?

Git не хранит и не обрабатывает данные таким же способом как другие СКВ.

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.

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

1) Зафиксированный значит, что файл уже сохранён в вашей локальной базе;

2) К изменённым относятся файлы, которые поменялись, но ещё не были зафиксированы;

3) Подготовленные файлы — это изменённые файлы, отмеченные для включения в следующий коммит.

7. Что такое профиль пользователя в GitHub?

Профиль - это наша публичная страница на GitHub, как и в социальных сетях. В нем другие пользователи могут посмотреть ваши работы.

8. Какие бывают репозитории в GitHub?

9. Укажите основные этапы модели работы с GitHub.

- 1) Регистрация;
- 2) Создание репозитория;
- 3) Клонирование репозитория;
- 4) Добавление новых файлов.

10. Как осуществляется первоначальная настройка Git после установки?

Убедимся, что Git установлен используя команду: `git version`. Перейдём в папку с локальным репозиторием используя команду: `cd /d <Расположения папки на компьютере>`. Свяжем локальный репозиторий и удалённый командами: `git config --global user.name <YOUR_NAME>` `git config --global user.email <EMAIL>`.

11. Опишите этапы создания репозитория в GitHub.

1) В правом верхнем углу, рядом с аватаром есть кнопка с плюсиком, нажимая которую мы переходим к созданию нового репозитория;

2) В результате будет выполнен переход на страницу создания репозитория. Наиболее важными на ней являются следующие поля: Имя репозитория. Описание (Description). Public/private. “Initialize this repository with a README” .gitignore и LICENSE.

12. Какие типы лицензий поддерживаются GitHub при создании репозитория?

Microsoft Reciprocal License, The Code Project Open License (CPO), The Common Development and Distribution License (CDDL), The Microsoft Public License (Ms-PL), The Mozilla Public License 1.1 (MPL 1.1), The Common Public License Version 1.0 (CPL), The Eclipse Public License 1.0, The MIT License, The BSD License, The Apache License, Version 2.0, The Creative Commons Attribution-ShareAlike 2.5 License, The zlib/libpng License, A Public Domain dedication, The Creative Commons Attribution 3.0 Unported License, The Creative Commons).

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

После создания репозитория его необходимо клонировать на ваш компьютер. Для этого на странице репозитория необходимо найти кнопку Clone или Code и щелкнуть по ней, чтобы отобразить адрес репозитория для клонирования.

Откройте командную строку или терминал и перейдите в каталог, куда вы хотите скопировать хранилище. Затем напишите `git clone` и введите адрес.

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` ?

Файлы обновятся на репозитории.

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.

`git clone.`

`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.

1) GitLab — альтернатива GitHub номер один. GitLab предоставляет не только веб-сервис для совместной работы, но и программное обеспечение с открытым исходным кодом;

2) BitBucket — это служба хостинга репозитория и управления версиями от Atlassian. Она тесно интегрирована с другими инструментами Atlassian — Jira, HipChat и Confluence.

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 это совершенно бесплатное приложение с открытым исходным кодом, разработанное GitHub. С его помощью можно взаимодействовать с GitHub (что и не удивительно), а также с другими платформами (включая Bitbucket и GitLab).

Вывод: исследовал базовые возможности системы контроля версий Git и веб-сервиса для хостинга IT-проектов GitHub.