

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 РАБОТЕ №1.1.
дисциплины «Основы кроссплатформенного программирования»

Выполнил:
Пазенко Данила Сергеевич
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., канд. тех. наук, доцент,
доцент кафедры инфокоммуникаций

(подпись)

Отчет защищен с оценкой _____ Дата защиты _____

Ставрополь, 2023 г.

Тема: исследование основных возможностей Git и GitHub.

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

Порядок выполнения работы:

1. Создание нового репозитория в GitHub.

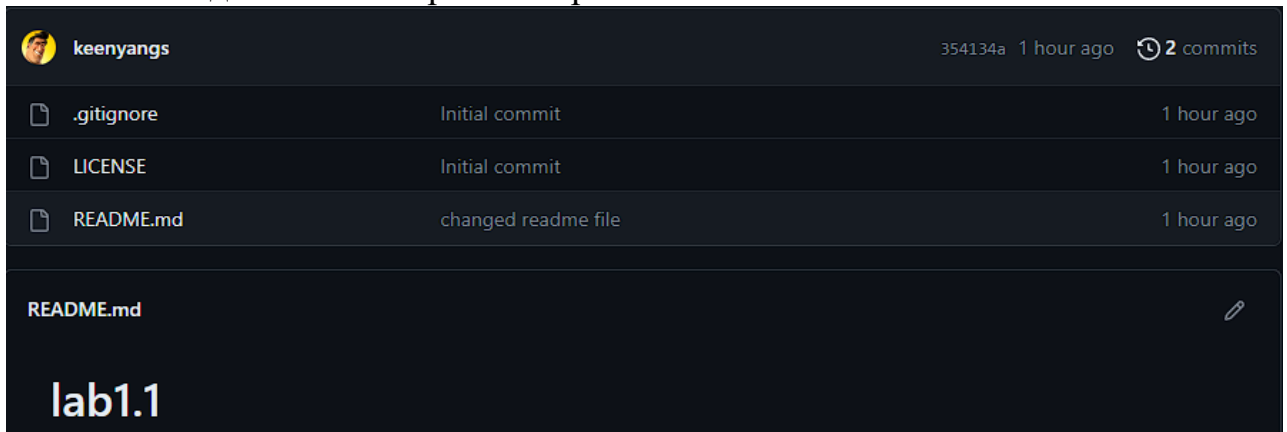


Рисунок 1. Новый репозиторий.

2. Ввел в командную строку `git -v`, таким образом проверил, что все работает.

```
stron@KPA3 MINGW64 /v/Programms/Project/lab1.1 (main)
$ git -v
git version 2.42.0.windows.1
```

Рисунок 2. git version.

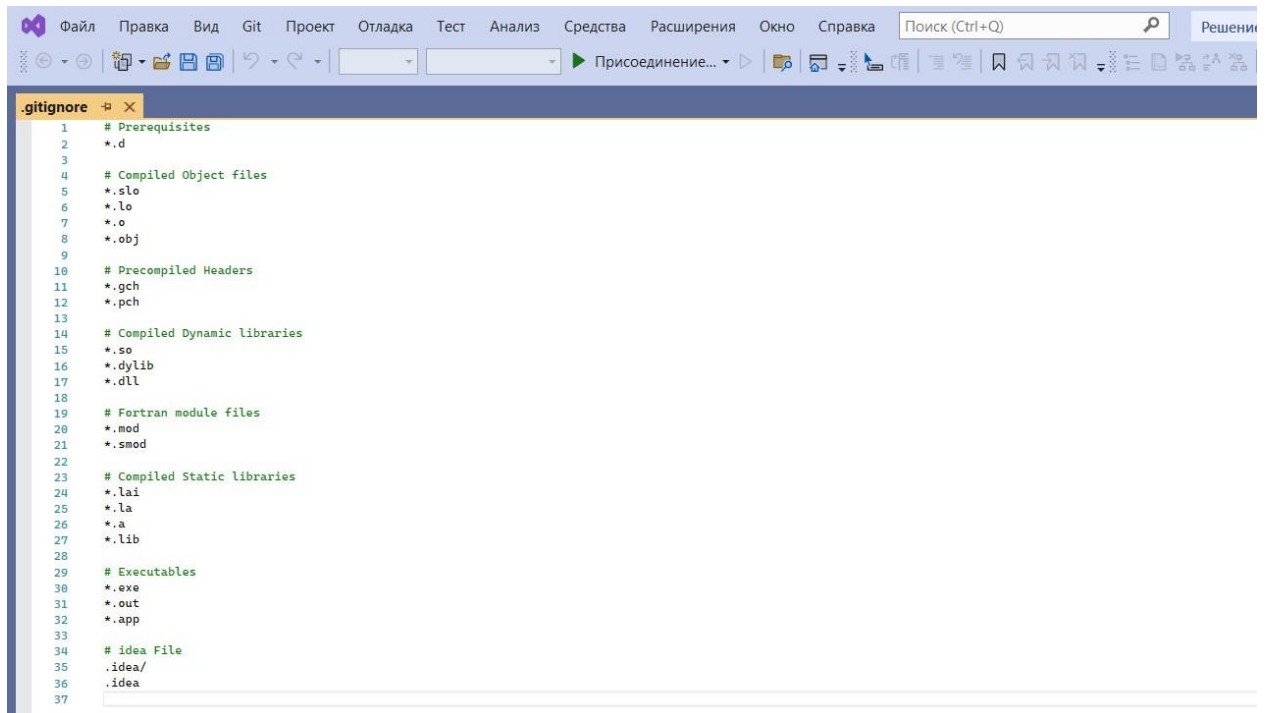
3. Ввел свое имя и свой email.

```
stron@KPA3 MINGW64 /v/Programms/Project/lab1.1 (main)
$ git config --global user.email "strong.pazenko@mail.ru"

stron@KPA3 MINGW64 /v/Programms/Project/lab1.1 (main)
$ git config --global user.name "Danila"
```

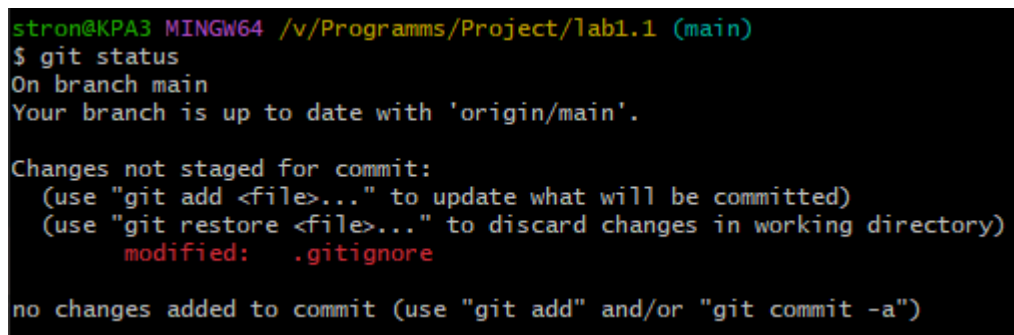
Рисунок 3. Имя и почта.

4. Дополнил файл .gitignore необходимым правилом игнорировать файлы .idea



```
.gitignore
1  # Prerequisites
2  *.d
3
4  # Compiled Object files
5  *.slo
6  *.lo
7  *.o
8  *.obj
9
10 # Precompiled Headers
11 *.gch
12 *.pch
13
14 # Compiled Dynamic Libraries
15 *.so
16 *.dylib
17 *.dll
18
19 # Fortran module files
20 *.mod
21 *.smod
22
23 # Compiled Static Libraries
24 *.lai
25 *.la
26 *.a
27 *.lib
28
29 # Executables
30 *.exe
31 *.out
32 *.app
33
34 # idea File
35 .idea/
36 .idea
37
```

Рисунок 6. Дополнение файла gitignore.



```
stron@KPA3 MINGW64 /v/Programms/Project/lab1.1 (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:   .gitignore

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

Рисунок 7. Изменение файла gitignore.

5. Внес изменения в файл README (имя и фамилию, группу).

```
# lab1.1
Pazenko Danila, ITS-bo-22-1, Osnovi krossplatformennogo programmirovaniya
```

Рисунок 8. Изменение файла README.

6. Написал небольшую программу на языке C++, фиксировал изменения при написании в локальном репозитории, сделал не менее 7 коммитов. Все это в файле README, как указано в методических указаниях.

```
commit ff921e3247b83e211beb9938f10c167e699b0a71 (HEAD -> main)
Author: Danila <strong.pazenko@mail.ru>
Date: Sat Aug 26 17:37:04 2023 +0300

    7 commit

commit c8fadc84f9dced6cb63d74c6c10cc0815d988837
Author: Danila <strong.pazenko@mail.ru>
Date: Sat Aug 26 17:36:30 2023 +0300

    6 commit

commit 707a9b5f49ad65605034f2fac705d07d14de3661
Author: Danila <strong.pazenko@mail.ru>
Date: Sat Aug 26 17:36:08 2023 +0300

    5 commit

commit 13546db18a239be53555d58b12049050c633ff49
Author: Danila <strong.pazenko@mail.ru>
Date: Sat Aug 26 17:35:33 2023 +0300

    4 commit

commit 8450f16037eebbd57917135b4710062a299f2eda
Author: Danila <strong.pazenko@mail.ru>
Date: Sat Aug 26 17:35:13 2023 +0300

    3 commit

commit f27ab8113a1c5e4a0382f1b2bd843f678af706c3
Author: Danila <strong.pazenko@mail.ru>
Date: Sat Aug 26 17:34:26 2023 +0300

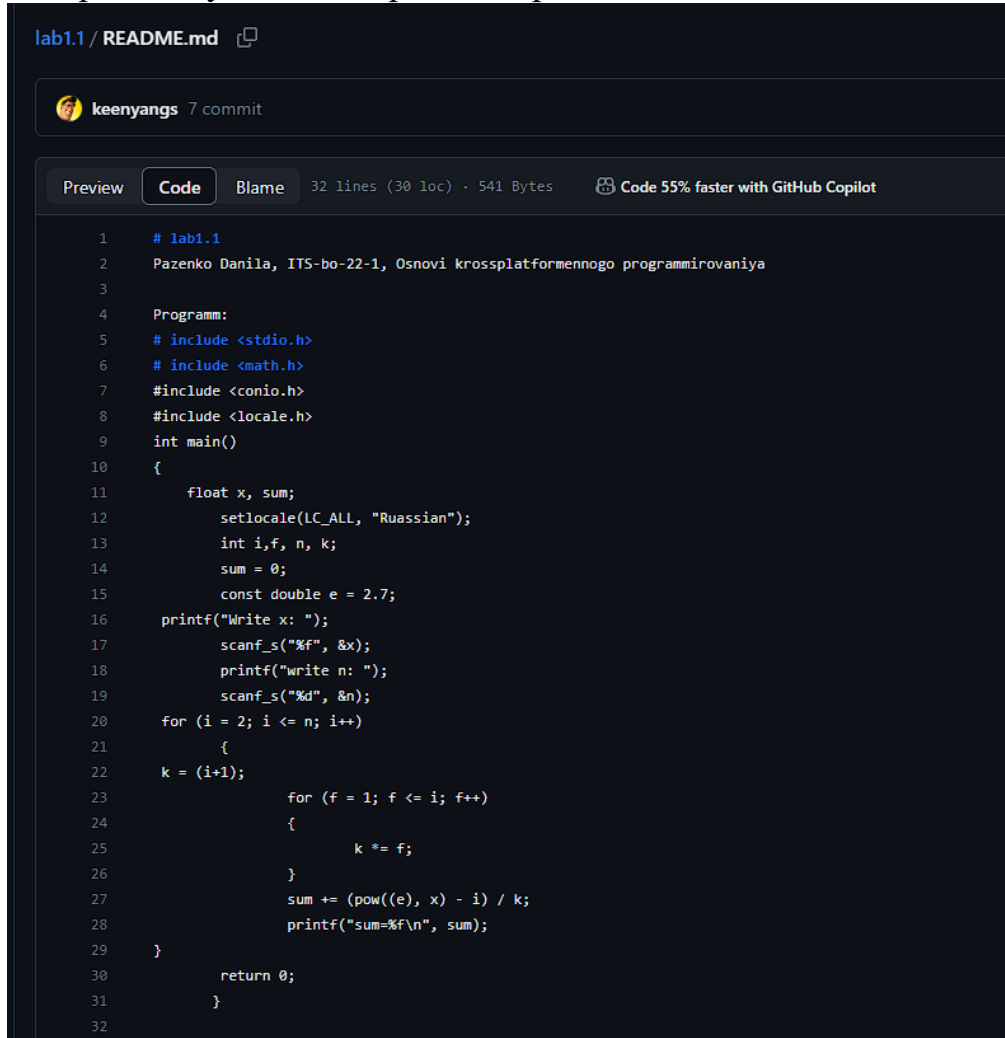
    2 commit

commit a36b24e704c9f2ed40b1c7dc6f1f6887e017473b
Author: Danila <strong.pazenko@mail.ru>
Date: Sat Aug 26 17:33:39 2023 +0300

    1 commit
```

Рисунок 9. Список коммит

7. Отправил в удаленный репозиторий GitHub.



The screenshot shows the GitHub interface for a repository named 'lab1.1'. The file 'README.md' is selected, showing a commit by user 'keenyangs' with 7 commits. The code is displayed in a dark-themed editor. The code is a C program that calculates the sum of a series. It includes headers for stdio, math, conio, and locale. It sets the locale to Russian and prompts the user to enter a value 'x'. It then calculates the sum of the series $\sum_{i=2}^n \frac{e^x - i}{k}$, where $k = (i+1)$. The program prints the result and returns 0.

```
1  # lab1.1
2  Pazenko Danila, ITS-bo-22-1, Osnovi krossplatformennogo programirovaniya
3
4  Programm:
5  # include <stdio.h>
6  # include <math.h>
7  #include <conio.h>
8  #include <locale.h>
9  int main()
10 {
11     float x, sum;
12     setlocale(LC_ALL, "Ruassian");
13     int i,f, n, k;
14     sum = 0;
15     const double e = 2.7;
16     printf("Write x: ");
17     scanf_s("%f", &x);
18     printf("write n: ");
19     scanf_s("%d", &n);
20     for (i = 2; i <= n; i++)
21     {
22         k = (i+1);
23         for (f = 1; f <= i; f++)
24         {
25             k *= f;
26         }
27         sum += (pow((e), x) - i) / k;
28         printf("sum=%f\n", sum);
29     }
30     return 0;
31 }
32
```

Рисунок 10. Проверка изменений в GitHub.

Ссылка: <https://github.com/keenyangs/lab1.1>

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 с тем, чтобы в дальнейшем была возможность вернуться к определённым старым версиям этих файлов.

2) В чем недостатки локальных и централизованных СКВ?

Основной недостаток локальных СКВ — можно легко забыть, в какой директории мы находимся, и случайно изменить не тот файл или скопировать не те файлы, которые мы хотели.

Основной недостаток централизованных СКВ заключается в том, что 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?

Git относится к распределённым СКВ (РСКВ)

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.

Git не хранит и не обрабатывает данные таким способом. Вместо этого, подход Git к хранению данных больше похож на набор снимков миниатюрной файловой системы.

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.

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 наша публичная страница на GitHub, как и в социальных сетях.

8) Какие бывают репозитории в GitHub?

Репозиторий бывает трех видов: локальный, централизованный, распределенный.

9) Укажите основные этапы модели работы с GitHub.

GitHub содержит в себе два хранилища:

A) *upstream* - это оригинальный репозиторий проекта, который мы скопировали.

B) *origin* - ваш fork (копия) на GitHub, к которому у вас есть полный доступ.

Чтобы перенести изменения с вашей копии в исходному репозиторий проекта, нам нужно сделать запрос на извлечение.

10) Как осуществляется первоначальная настройка Git после установки?

Чтобы убедиться в том, что мы установили Git правильно необходимо вписать команду *git version*, если она сработала необходимо написать свое имя и почту с помощью следующих команд:

```
git config --global user.name "Name"
```

git config --global user.email "Email"

11) Опишите этапы создания репозитория в GitHub.

- a) *Имя репозитория*. Оно может быть любое, необязательно уникальное во всем github, потому что привязано к вашему аккаунту, но уникальное в рамках тех репозиториях, которые вы создавали.
- b) *Описание (Description)*. Можно оставить пустым.
- c) *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).
- d) *.gitignore и LICENSE* можно сейчас не выбирать.

12) Какие типы лицензий поддерживаются GitHub при создании репозитория?

- a) Лицензия Apache 2.0;
- b) MIT License;
- c) Публичная лицензия Eclipse 2.0;
- d) GNU Affero General Public License 2.0;

И многие другие.

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

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 скопированный адрес.

14) Как проверить состояние локального репозитория Git?

Проверить состояние локального репозитория можно с помощью команды *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 файла в локальных репозиторий Git состояние локального репозитория измениться на `modified` – измененное.

При добавлении нового/изменного файла под версионный контроль состояние локального репозитория измениться на `staged` – подготовленное.

При фиксации и отправки изменений на сервер состояние перейдет в `committed` – зафиксированное.

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` .

Для получения обновлений с удаленного репозитория можно воспользоваться командой: *`git pull`*.

Если вы изменили ваши локальные файлы, то команда `git pull` выдаст ошибку. Если вы уверены, что хотите перезаписать локальные файлы, файлами из удаленного репозитория то выполните команды:

`git fetch --all`

`git reset --hard github/master`

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

Сервисы работающие с Git:

- a) Fork;
- b) Tower;

- c) Sourcetree;
- d) SmartGit;
- e) GitKraken.

Сравню сервис Fork с GitHub. В фокусе этого инструмента скорость, дружелюбность к пользователю и эффективность. К особенностям Fork можно отнести красивый вид, кнопки быстрого доступа, встроенную систему разрешения конфликтов слияния, менеджер репозитория. Основная его черта – скорость и простота для пользователя.

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.

Существует и другое программное средство с графическим интерфейсом, например, Git GUI – предназначен для тех, кто не любит командную строку.

Для создания локального репозитория: в нашем графическом интерфейсе Git нажмите “Создать новый репозиторий”.

Выбрать местоположение, в котором вы хотите сохранить свой репозиторий.

Чтобы клонировать репозиторий, нажмите на ссылку “Клонировать существующий репозиторий” в окне Git GUI.

Существующий репозиторий - это тот, который уже инициализирован и / или имеет отправленные в него коммиты.

Когда мы перемещаем файлы в каталог Git, вы увидите все файлы в окне “Неустановленные изменения”. Это в основном означает, что новые файлы были добавлены, удалены, обновлены и т.д.

Когда мы нажимаем кнопку “Этап изменен”, он попытается добавить все новые файлы в индекс Git.

Так осуществляются похожие действия в Git GUI, которые были описаны в лабораторной работе.

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