

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 №2
дисциплины «Основы кроссплатформенного программирования»

Выполнил:

Кондратенко Даниил Витальевич

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.

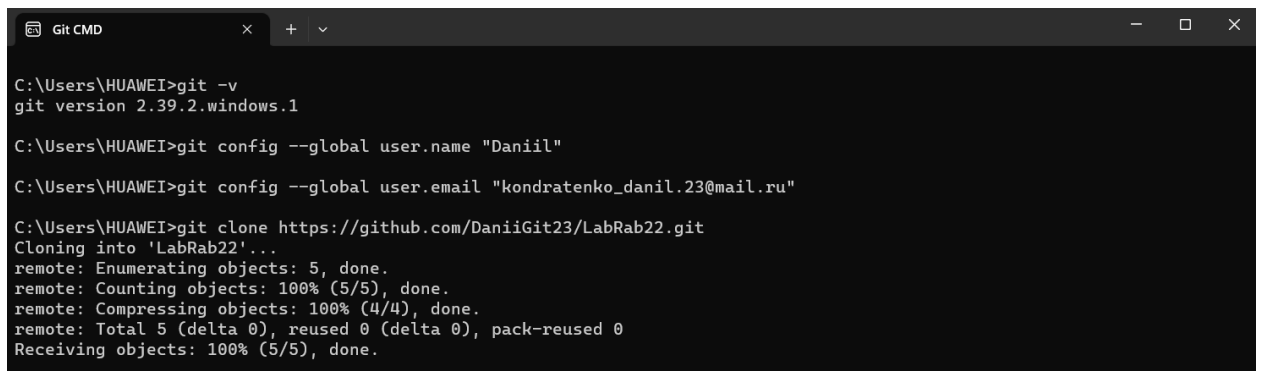
Тема: исследование возможностей Git для работы с локальными репозиториями.

Цель работы: исследовать базовые возможности системы контроля версий Git для работы с локальными репозиториями.

Порядок выполнения работы:

Задание 1.

Создал новый репозиторий и клонировал его на свой компьютер.



```
C:\Users\HUAWEI>git -v
git version 2.39.2.windows.1

C:\Users\HUAWEI>git config --global user.name "Daniil"

C:\Users\HUAWEI>git config --global user.email "kondratenko_danil.23@mail.ru"

C:\Users\HUAWEI>git clone https://github.com/DaniiGit23/LabRab22.git
Cloning into 'LabRab22'...
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 5 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (5/5), done.
```

Рисунок 1. Новый репозиторий

Задание 2.

Добавил некоторое правило в файл *gitignore*, чтобы Git игнорировал файлы в формате *.swp*

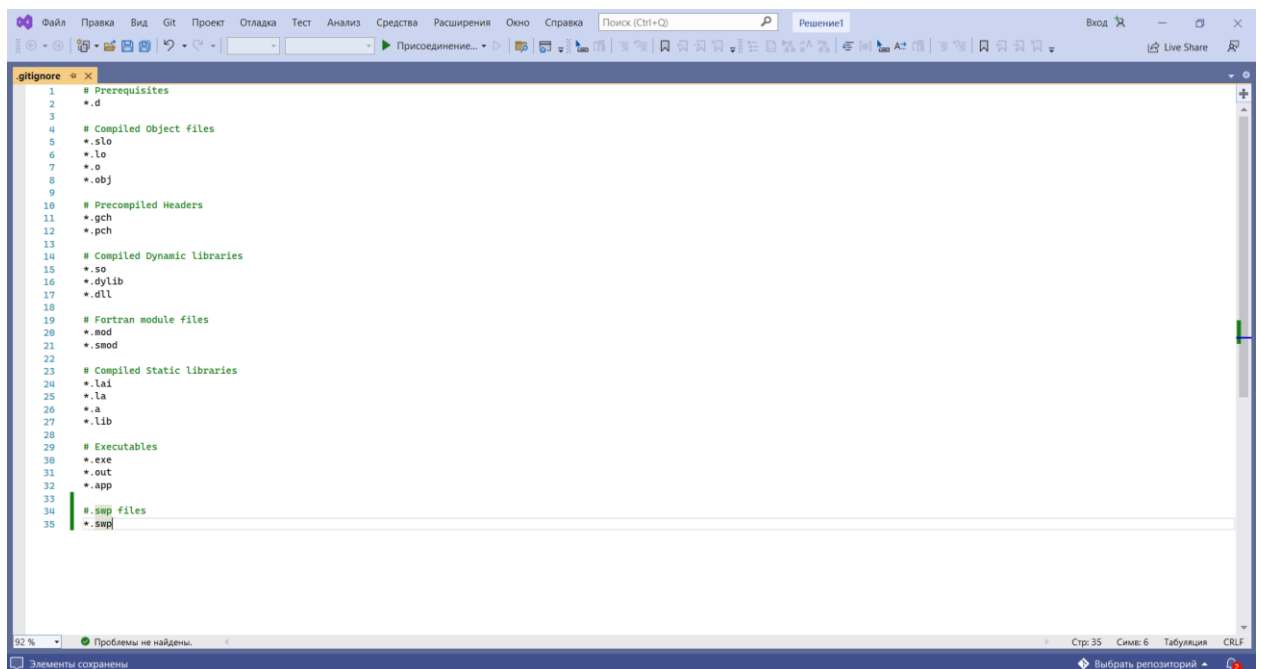


Рисунок 2. Работа с *gitignore*

Задание 3.

Добавил информацию в файл README.md о дисциплине, группе и ФИО.

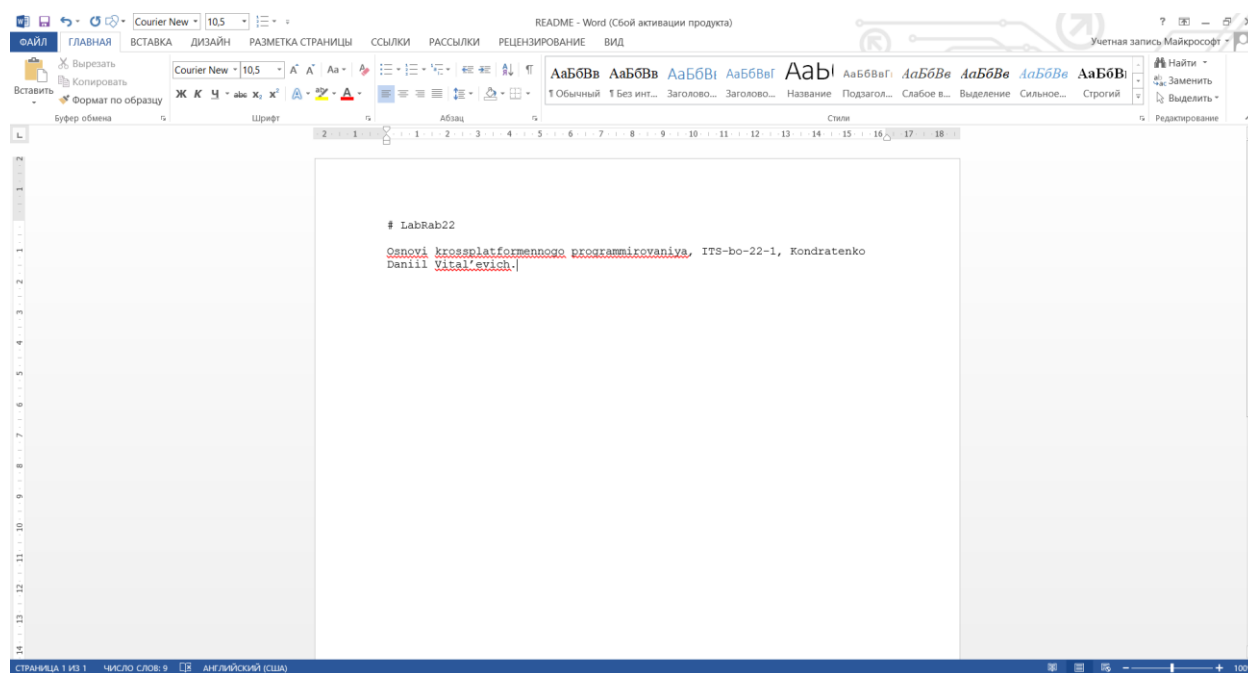


Рисунок 3. Работа с README

Задание 4.

Написал небольшую программу в новом файле `main.cpp`, сделал не менее 7-ми коммитов с 3-мя тегами.

```
C:\Users\HUAWEI\LabRab22>git log --graph --pretty=oneline --abbrev-commit
* 66d9bbc (HEAD -> main, tag: konec) Konec
* a3595f8 Yslovie inache
* 7904db3 Ina4e
* a255677 telo ysloviya
* 85e5ca0 Yslovie
* 19737b6 Formyla
* 2cc45b0 (tag: disreminant) Vvod peremennih
* 2e03de2 Nachalo
* 2162f0d (tag: Nachalo, origin/main, origin/HEAD) Initial commit

C:\Users\HUAWEI\LabRab22>
```

Рисунок 4. История хранилища

С помощью команды `git log --graph --pretty=oneline --abbrev-commit` можно проанализировать все свои коммиты и теги. `--graph` отображает ASCII граф с ветвлениями и историей слияний, `--pretty=oneline` показывает коммиты в альтернативном формате и `--abbrev-commit` показывает только несколько символов SHA-1 чек-суммы вместо всех 40.

Задание 5.

Посмотрел содержимое коммитов командой `git show <ref>`, где <ref>:

- 1) HEAD : последний коммит;

```
C:\Users\HUAWEI\LabRab22>git show HEAD
commit 66d9bbcd2651911713ba7bfad3a38f83c4c24a8 (HEAD -> main, tag: konec)
Author: Daniil <kondratenko_danil.23@mail.ru>
Date: Tue Mar 14 18:16:43 2023 +0300

    Konec

diff --git a/main.cpp b/main.cpp
index 9e0aa62..03ee6de 100644
--- a/main.cpp
+++ b/main.cpp
@@ -25,3 +25,7 @@ else
         x3 = (-b) / (2 * a);
         printf("%lf\n", x3);
     }
+else
+printf("Net sasheniy");
+}
+    }
```

Рисунок 5. Последний коммит

- 2) HEAD~1 : предпоследний коммит (и т. д.);

```
C:\Users\HUAWEI\LabRab22>git show HEAD~1
commit a3595f88b48ffc82ab1439b09673be1125e43510
Author: Daniil <kondratenko_danil.23@mail.ru>
Date: Tue Mar 14 18:15:20 2023 +0300

    Yslovie inache

diff --git a/main.cpp b/main.cpp
index eed9953..9e0aa62 100644
--- a/main.cpp
+++ b/main.cpp
@@ -18,4 +18,10 @@ if (D > 0)
     printf("%lf\n", x1);
     printf("%lf\n", x2);
 }
-else
+{
+    if (D == 0)
+    {
+        x3 = (-b) / (2 * a);
+        printf("%lf\n", x3);
+    }
\ No newline at end of file
+else
+{
+    if (D == 0)
+    {
+        x3 = (-b) / (2 * a);
+        printf("%lf\n", x3);
+    }
+}
```

Рисунок 6. Предпоследний коммит.

3) b34a0e : коммит с указанным хэшем.

```
C:\Users\HUAWEI\LabRab22>git show 19737b6
commit 19737b67ae1f005c41c76a1415f9bee31c6b4307
Author: Daniil <kondratenko_danil.23@mail.ru>
Date:   Tue Mar 14 18:11:26 2023 +0300

    Formyla

diff --git a/main.cpp b/main.cpp
index 750d559..50593ea 100644
--- a/main.cpp
+++ b/main.cpp
@@ -10,3 +10,4 @@ printf("Write b ");
    scanf_s("%lf", &b);
    printf("Write c ");
    scanf_s("%lf", &c);
+D = b * b - 4 * a * c;
\ No newline at end of file

C:\Users\HUAWEI\LabRab22>
```

Рисунок 7. Коммит с указанным хэшем.

Задание 6. Откат к заданной версии.

1.1. Удалил весь программный код с файла main.cpp и сохранил его.

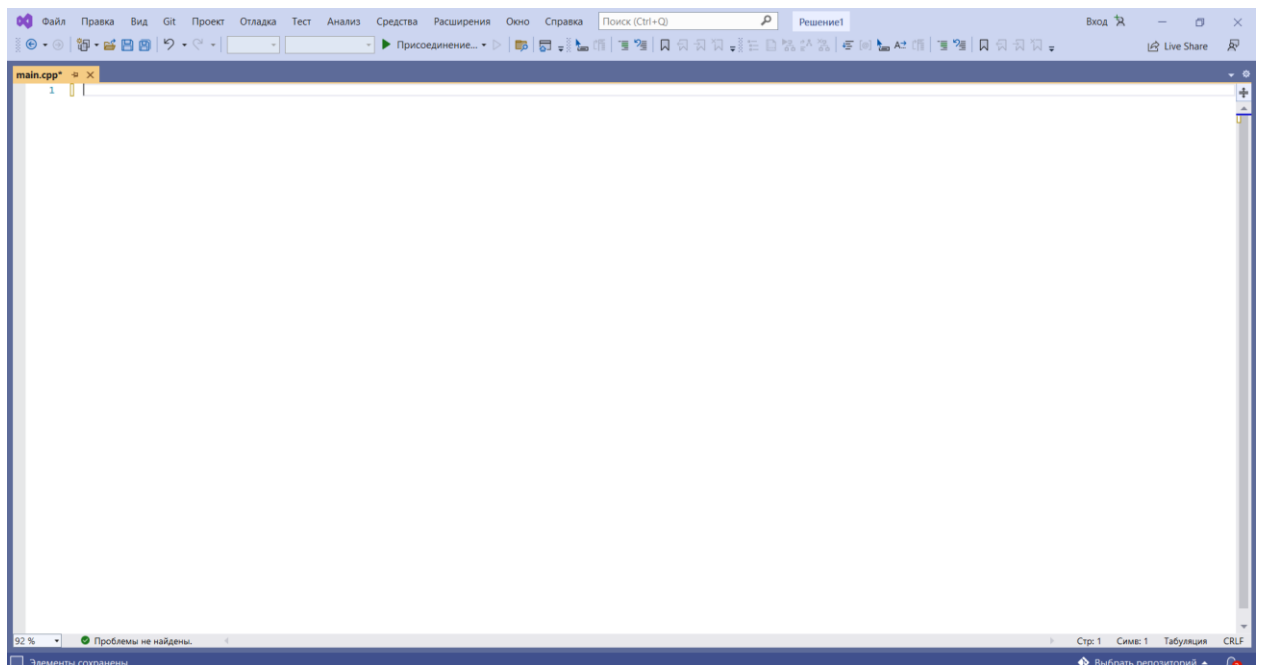


Рисунок 8. Удаление программ

1.2. Удалил это изменение с помощью команды *git checkout -- main.cpp*.

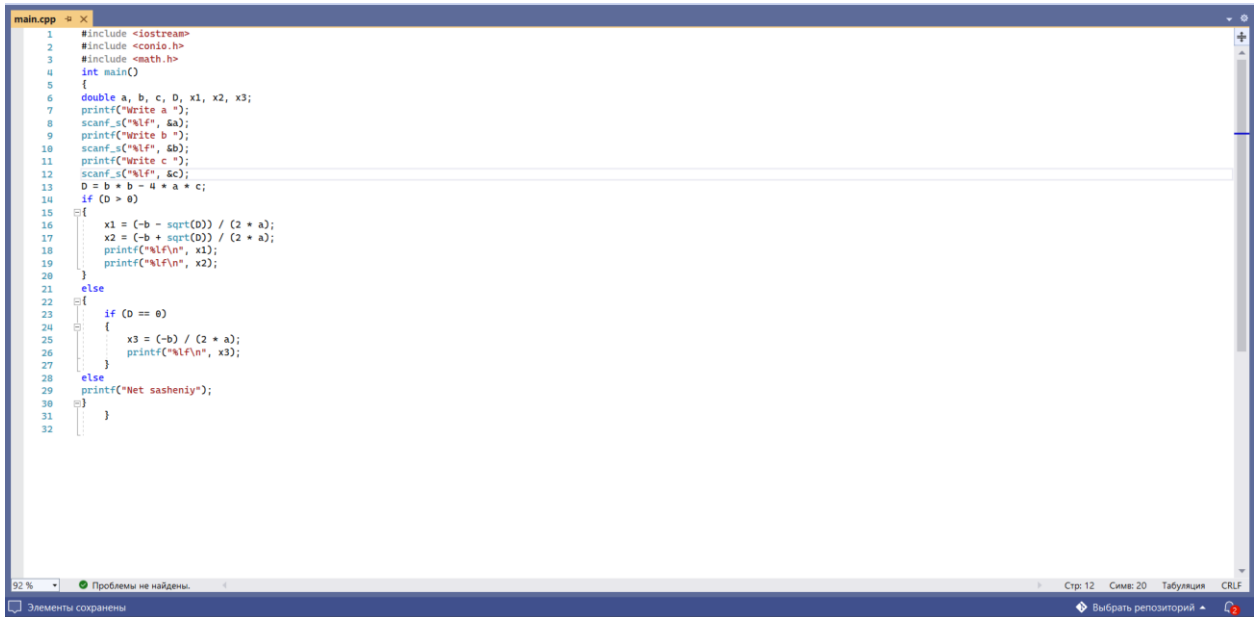


Рисунок 9. Восстановление программы.

Код вновь вернулся.

1.3. Вновь повторил пункт 1.1. и сделал коммит.

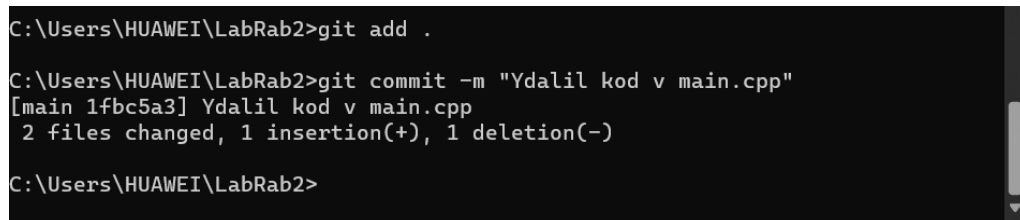


Рисунок 10. Коммит

1.4. Откатить состояние хранилища к предыдущей версии командой:
git reset --hard HEAD~1 .

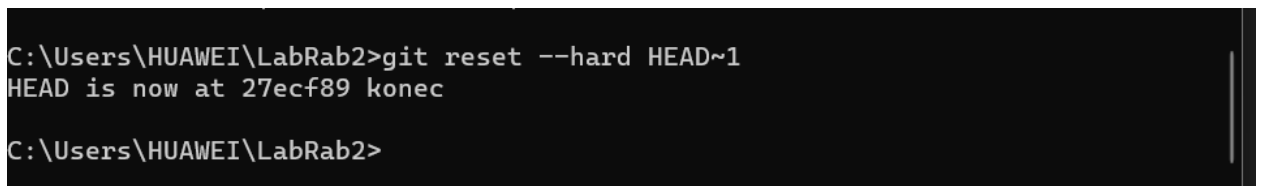


Рисунок 10. Возвращение к предпоследней версии коммита

Код вновь вернулся.

После сделанных пунктов можно сделать вывод, что можно отменять ненужные изменения, в случае если коммит не произошел и изменения не нужны. Также, если коммит был уже сделан, то можно вернуться к

предпоследней версии коммита (где было сохранение до сохранения с ненужными изменениями).

Ссылка на репозиторий: <https://github.com/DaniiGit23/LabRab22.git>

Ответы на контрольные вопросы:

1) Как выполнить историю коммитов в Git? Какие существуют дополнительные опции для просмотра истории коммитов?

Историю коммитов можно выполнить с помощью команды `git log`.

Дополнительные опции для просмотра истории:

`%H`, `%h`, `%T`, `%t`, `%P`, `%p` тд.

`-p`, `--stat`, `--shortstat`, `--name-only`, `--name-status` и тд.

2) Как ограничить вывод при просмотре истории коммитов?

Ограничить вывод при просмотре истории коммитов можно с помощью команды `git log -n`, где `n` — число последних коммитов.

3) Как внести изменения в уже сделанный коммит?

Если вы хотите переделать коммит — внесите необходимые изменения, добавьте их в индекс и сделайте коммит ещё раз, указав

параметр `--amend` : `git commit --amend`.

4) Как отменить индексацию файла в Git?

Отменить индексацию файла можно с помощью команды: `git reset HEAD <file>`.

5) Как отменить изменения в файле?

Отменить изменения в файле можно с помощью команды: `git checkout - <file>`

6) Что такое удаленный репозиторий Git?

Удалённые репозитории представляют собой версии вашего проекта, сохранённые в интернете или ещё где-то в сети.

7) Как выполнить просмотр удаленных репозиториях данного локального репозитория?

Выполнить просмотр удаленных репозиторий данного локального репозитория можно с помощью команды: *git remote*.

8) Как добавить удаленный репозиторий для данного локального репозитория?

Для того, чтобы добавить удалённый репозиторий и присвоить ему имя (shortname), просто выполните команду *git remote add <shortname> <url>*.

9) Как выполнить отправку/получение изменений с удаленного репозитория?

Для получения данных из удалённых проектов, следует выполнить: *git fetch [remote-name]*.

Для отправки изменений в удаленный репозиторий используется команда: *git push <remote-name> <branch-name>*

10) Как выполнить просмотр удаленного репозитория?

Если хотите получить побольше информации об одном из удалённых репозиторий, вы можете использовать команду: *git remote show <remote>*.

11) Каково назначение тэгов Git?

Git имеет возможность пометить определённые моменты в истории как важные. Для таких случаев были придуманы тэги.

12) Как осуществляется работа с тэгами Git?

Просмотреть список имеющихся тегов в Git можно очень просто. Достаточно набрать команду *git tag*.

Создание аннотированного тега в Git выполняется легко. Самый простой способ — это указать *-a* при выполнении команды *tag*.

С помощью команды *git show* вы можете посмотреть данные тега вместе с коммитом.

По умолчанию, команда *git push* не отправляет теги на удалённые сервера. После создания теги нужно отправлять явно на удалённый сервер. Процесс аналогичен отправке веток — достаточно выполнить команду *git push origin <tagname>*.

Для удаления тега в локальной репозитории достаточно выполнить команду `git tag -d <tagname>` .

Если вы хотите получить версии файлов, на которые указывает тег, то вы можете сделать `git checkout <tagname>` для тега.

13) Самостоятельно изучите назначение флага `--prune` в командах `git fetch` и `git push`. Каково назначение этого флага?

`Git prune` – это команда, которая удаляет все файлы, недоступные из текущей ветки. Команда `prune` полезна, когда в вашем рабочем каталоге много файлов, которые вы не хотите хранить.

`git fetch --prune` делает то же самое: удалит ссылки на ветки, которые не существуют на удаленном компьютере.

Опция `--prune` в команде `git push` удалит ветку из удаленного репозитория, если в локальной репозитории не существует ветки с таким именем.

Вывод: исследовал базовые возможности системы контроля версий `Git` для работы с локальными репозиториями.