

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 №1.2.
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 возможностей Git для работы с локальными репозиториями.

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

Порядок выполнения работы:

1. Создал новый репозиторий и клонировал его на свой компьютер.

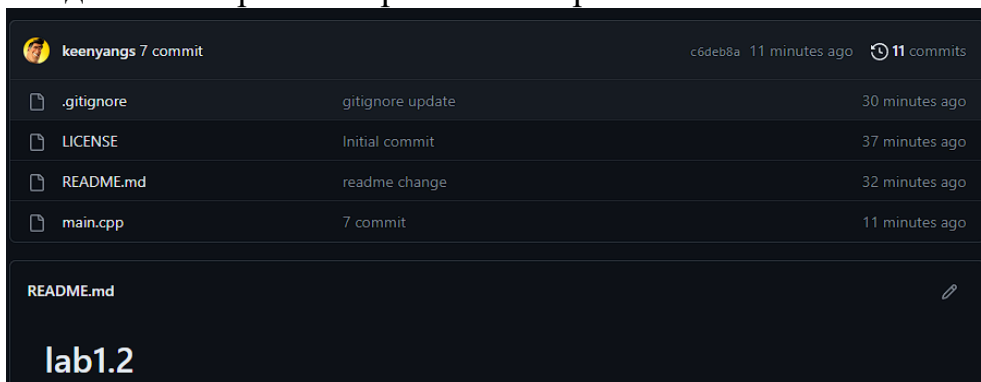


Рисунок 1. Новый репозиторий

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

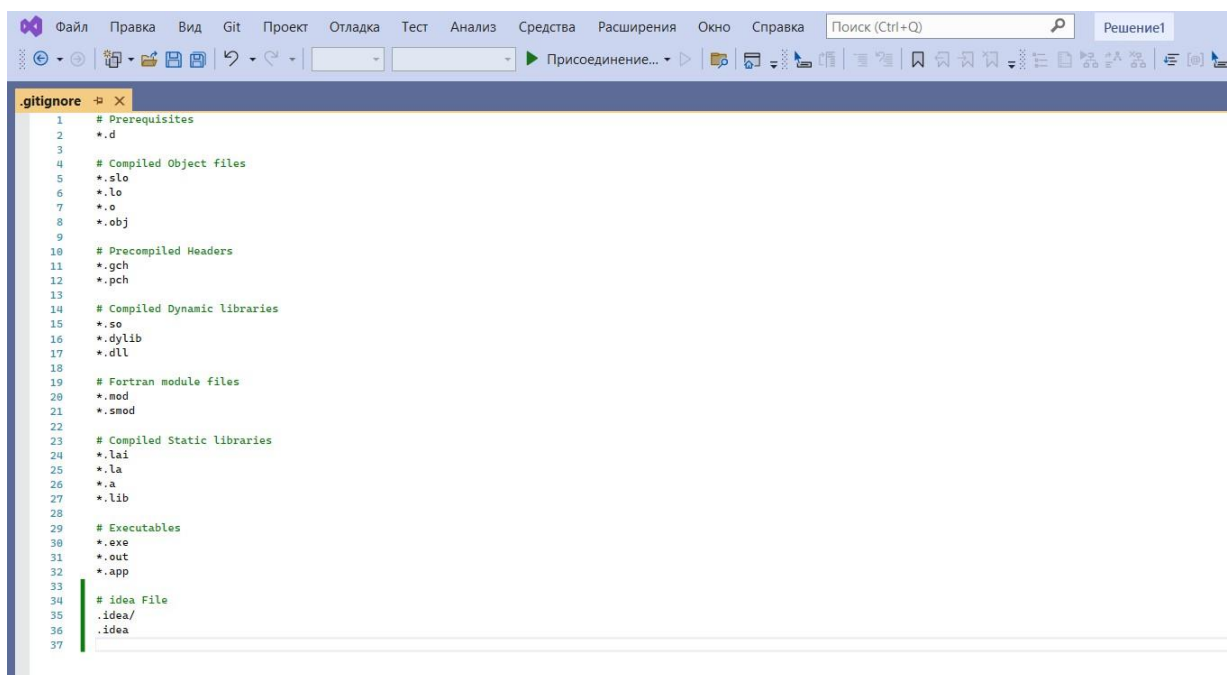


Рисунок 2. Работа с gitignore

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

```
# lab1.2
Pazenko Danila, ITS-bo-22-1, Osnovi krossplatformennogo programmirovaniya
```

Рисунок 3. Работа с README

4. Написал программу в новом файле main.cpp, сделал не менее 7-ми коммитов с 4-мя тегами.

```
commit c6deb8a44c6f62003dd6fab2b4deb9113a188284 (HEAD -> main, origin/main, origin/HEAD)
Author: Danila <strong.pazenko@mail.ru>
Date: Sat Aug 26 19:09:56 2023 +0300

    7 commit

commit c79ee73b5104cb7dc8098f7b46b5c7fb939f9641 (tag: v4)
Author: Danila <strong.pazenko@mail.ru>
Date: Sat Aug 26 19:08:41 2023 +0300

    6 commit

commit 61dc7f16e56d370549536544b7704d9999a333fa (tag: v3)
Author: Danila <strong.pazenko@mail.ru>
Date: Sat Aug 26 19:06:51 2023 +0300

    5 commit

commit aa9e62188117bc529de64d2e56b3baed600e660b
Author: Danila <strong.pazenko@mail.ru>
Date: Sat Aug 26 19:05:18 2023 +0300

    4 commit

commit 5408173759fa805973d28d8dd526eb41a36f4ff0
Author: Danila <strong.pazenko@mail.ru>
Date: Sat Aug 26 19:03:55 2023 +0300

    3 commit

commit 2c2a683b4efaaafbd0705c5b87b6e6708289e254a (tag: v2)
Author: Danila <strong.pazenko@mail.ru>
Date: Sat Aug 26 19:01:30 2023 +0300

    2 commit

commit 75eab08f138b06d1dcc005b94f8e03ce50ee9e53
Author: Danila <strong.pazenko@mail.ru>
Date: Sat Aug 26 18:56:32 2023 +0300

    1 commit

commit 6b23bf8d1bb601f64e6c178f16c9f7daaad6b6bef (tag: v1)
Author: Danila <strong.pazenko@mail.ru>
Date: Sat Aug 26 18:50:32 2023 +0300
```

Рисунок 4. История хранилища

Задание 5.

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

1) HEAD : последний коммит;

```
strong@KPA3 MINGW64 /v/Programms/Project/lab1.2 (main)
$ git show HEAD
commit c6deb8a44c6f62003dd6fab2b4deb9113a188284 (HEAD -> main, origin/main, origin/HEAD)
Author: Danila <strong.pazenko@mail.ru>
Date: Sat Aug 26 19:09:56 2023 +0300

    7 commit

diff --git a/main.cpp b/main.cpp
index 561df14..02d4a9a 100644
--- a/main.cpp
+++ b/main.cpp
@@ -24,4 +24,7 @@ int main()
     {
         x3 = (-b) / (2 * a);
         printf("%lf\n", x3);
-    }
+    }
+    else printf("Net sasheniy");
+}
\ No newline at end of file
```

Рисунок 5. Последний коммит

2) HEAD~1 : предпоследний коммит.

```
stron@KPA3 MINGW64 /v/Programms/Project/lab1.2 (main)
$ git show HEAD~1
commit c79ee73b5104cb7dc8098f7b46b5c7fb939f9641 (tag: v4)
Author: Danila <strong.pazenko@mail.ru>
Date: Sat Aug 26 19:08:41 2023 +0300

    6 commit

diff --git a/main.cpp b/main.cpp
index d69ef75..561df14 100644
--- a/main.cpp
+++ b/main.cpp
@@ -17,4 +17,11 @@ int main()
     x2 = (-b + sqrt(D)) / (2 * a);
     printf("%lf\n", x1);
     printf("%lf\n", x2);
-
+    }
+    \ No newline at end of file
+    }
+    else
+    {
+        if (D == 0)
+        {
+            x3 = (-b) / (2 * a);
+            printf("%lf\n", x3);
+        }
+    }
+    \ No newline at end of file
```

Рисунок 6. Предпоследний коммит.

3) 2c2a68: коммит с указанным хэшем.

```
stron@KPA3 MINGW64 /v/Programms/Project/lab1.2 (main)
$ git show 2c2a68
commit 2c2a683b4efaafbd0705c5b87b6e6708289e254a (tag: v2)
Author: Danila <strong.pazenko@mail.ru>
Date: Sat Aug 26 19:01:30 2023 +0300

    2 commit

diff --git a/main.cpp b/main.cpp
index 3964a14..2323eca 100644
--- a/main.cpp
+++ b/main.cpp
@@ -1,3 +1,5 @@
#include <iostream>
#include <conio.h>
#include <math.h>
+int main()
+{
+
+    \ No newline at end of file
```

Рисунок 7. Коммит с указанным хэшем.

6. Откат к заданной версии.

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

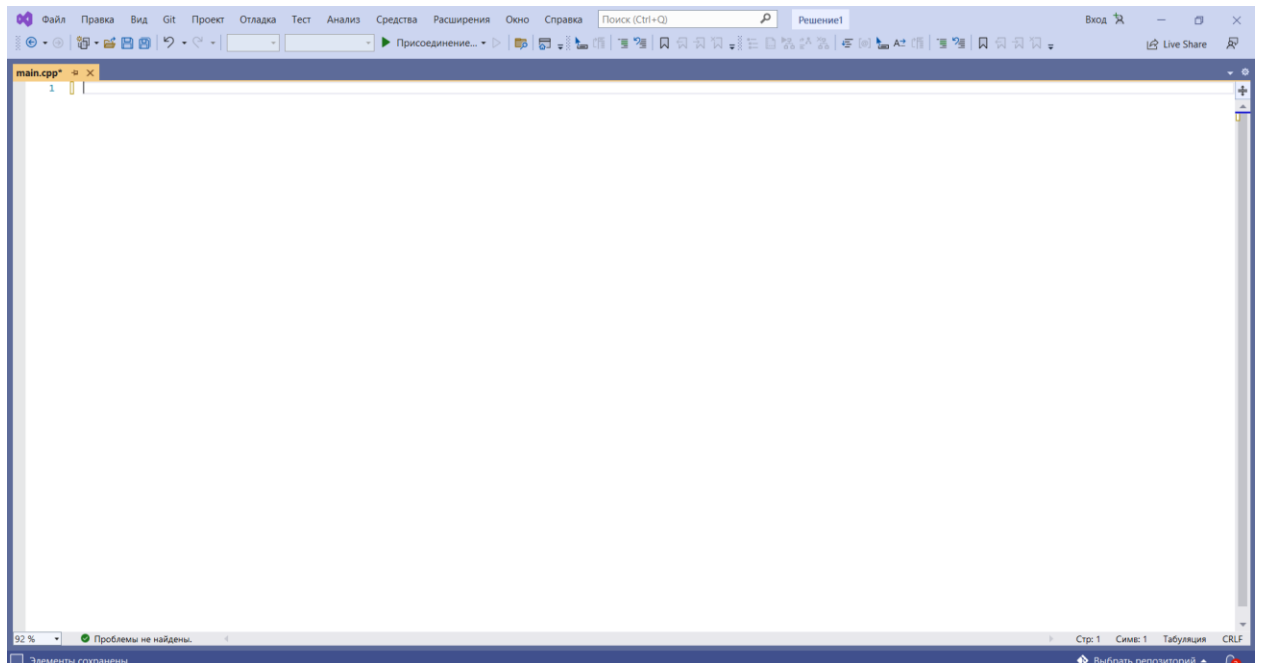


Рисунок 8. Удаление программы

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

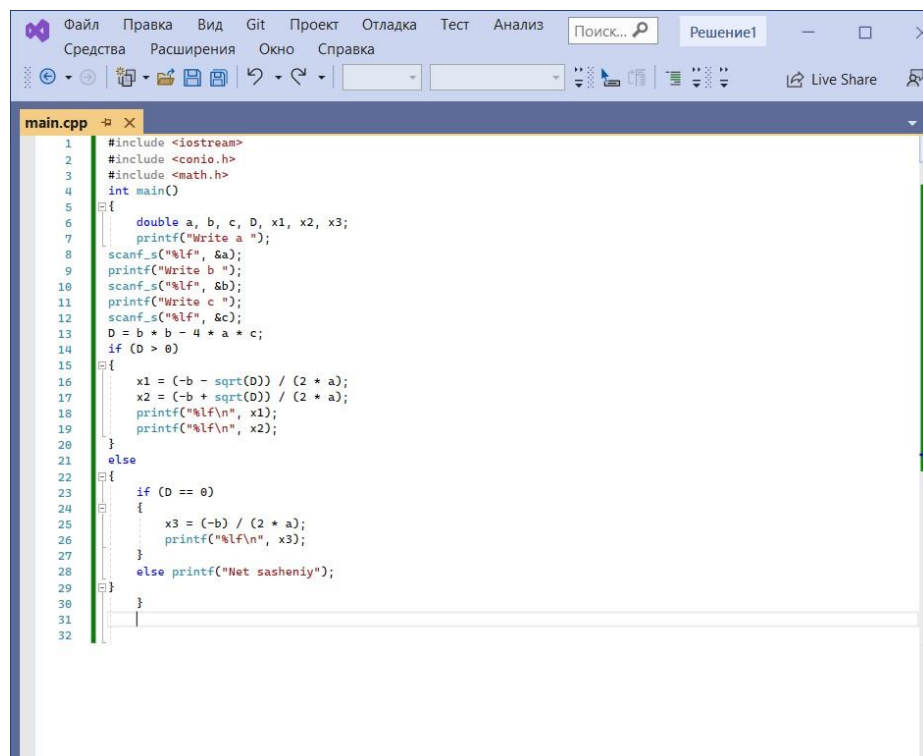


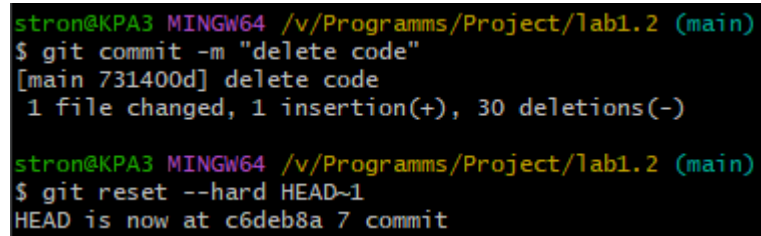
Рисунок 9. Восстановление программы.

Код вновь вернулся.

1.3. Вновь повторил пункт 1.1. и сделал коммит.

1.4. Откатить состояние хранилища к предыдущей версии командой:

git reset --hard HEAD~1 .



```
stron@KPA3 MINGW64 /v/Programms/Project/lab1.2 (main)
$ git commit -m "delete code"
[main 731400d] delete code
1 file changed, 1 insertion(+), 30 deletions(-)

stron@KPA3 MINGW64 /v/Programms/Project/lab1.2 (main)
$ git reset --hard HEAD~1
HEAD is now at c6deb8a 7 commit
```

Рисунок 10. Возвращение к предпоследней версии коммита

Код вновь вернулся.

Ссылка: <https://github.com/keenyangs/lab1.2>

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 репозиториях 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 репозиториях, 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.