

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:
Липчанский Дмитрий 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.

```
C:\Work>git clone https://github.com/Dmitry-3556/lab1.2.git
Cloning into 'lab1.2'...
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.

C:\Work>
```

Рисунок 1. Новый репозиторий

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

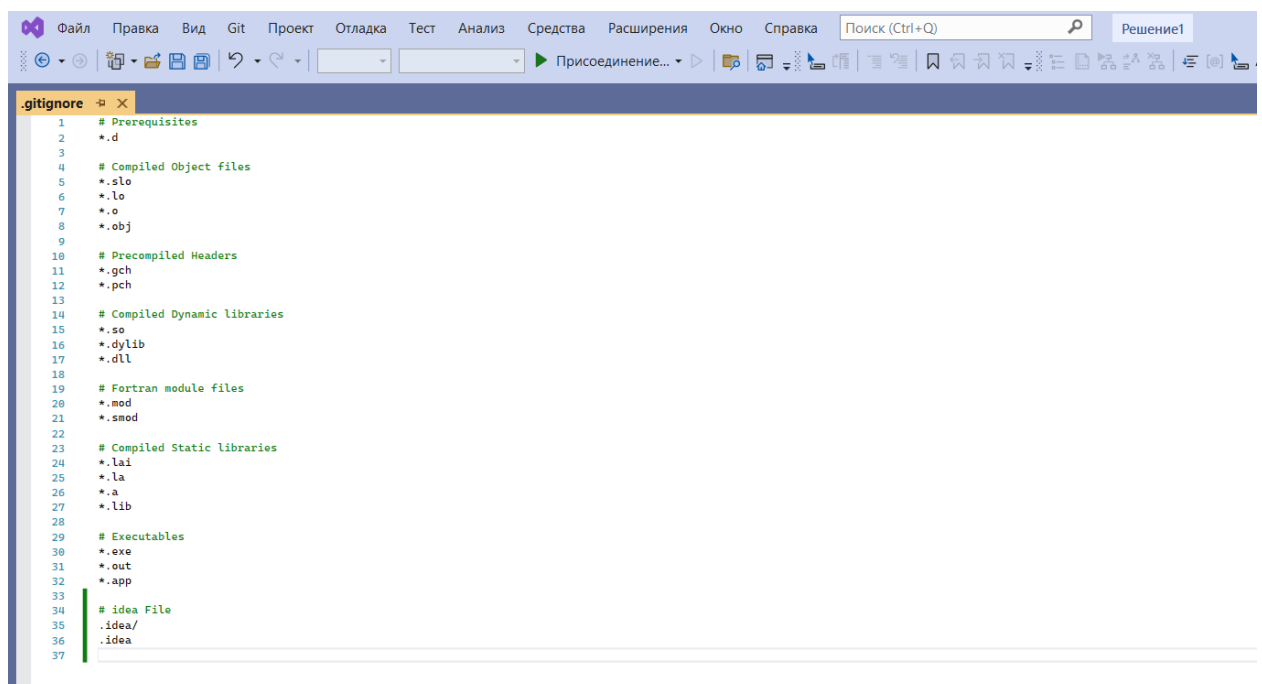


Рисунок 2. Работа с gitignore

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

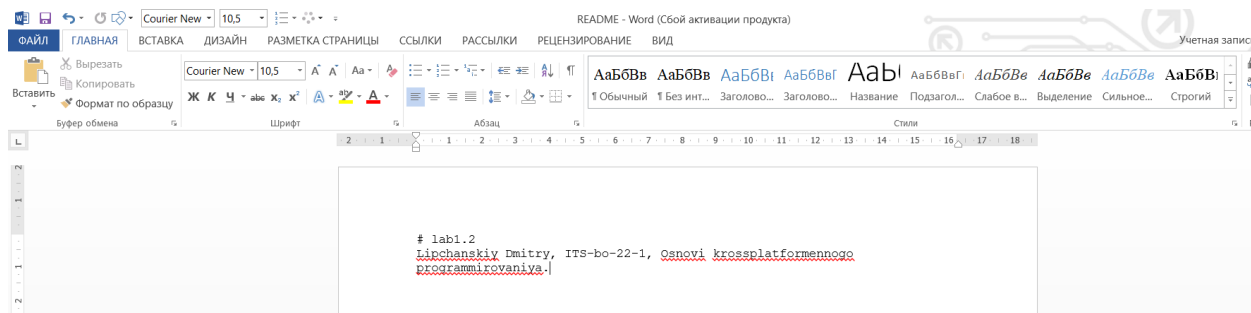


Рисунок 3. Работа с README

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

```
commit 49e4dce9f4696b0498203b0fe9b40ffcf5670af4 (HEAD -> main, tag: konecc)  
Author: Dmitry <heisenberg3556poul@gmail.com>  
Date: Mon May 29 13:40:22 2023 +0300  
  
7 commit  
  
commit 38f749d396e61a6b602957646d6f02542494819a (tag: konec)  
Author: Dmitry <heisenberg3556poul@gmail.com>  
Date: Mon May 29 13:38:04 2023 +0300  
  
6 commit  
  
commit 1f1c559a8a7cc14c1a57b00a9e97dffd13ce3552  
Author: Dmitry <heisenberg3556poul@gmail.com>  
Date: Mon May 29 13:37:46 2023 +0300  
  
5 commit  
  
commit 8ece19a7fd0618a020691a75aa2f6c1dbac405ec  
Author: Dmitry <heisenberg3556poul@gmail.com>  
Date: Mon May 29 13:37:09 2023 +0300  
  
4 commit  
  
commit a62814875c110397c77611de875060aaac287d50 (tag: v1)  
Author: Dmitry <heisenberg3556poul@gmail.com>  
Date: Mon May 29 13:32:52 2023 +0300  
  
3 commit  
  
commit efe3669ae0fd7a303765a0f8ed0b423850591d93  
Author: Dmitry <heisenberg3556poul@gmail.com>  
Date: Mon May 29 13:32:35 2023 +0300  
  
2 commit  
  
commit 9014ad863ccfa93a5060a461f90ebf7be43783 (tag: v)  
Author: Dmitry <heisenberg3556poul@gmail.com>  
Date: Mon May 29 13:31:20 2023 +0300  
  
1 commit
```

Рисунок 4. История хранилища

Задание 5.

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

- 1) HEAD : последний коммит;

```
C:\Work\lab1.2>git show HEAD
commit 49e4dce9f4696b0498203b0fe9b40ffcf5670af4 (HEAD -> main, tag: konecc)
Author: Dmitriy <heisenberg3556poul@gmail.com>
Date:   Mon May 29 13:40:22 2023 +0300

    7 commit

diff --git a/main.cpp b/main.cpp
index 1b0c047..65c5801 100644
--- a/main.cpp
+++ b/main.cpp
@@ -25,4 +25,7 @@ else
             x3 = (-b) / (2 * a);
             printf("%lf\n", x3);
         }
+    else printf("Net sasheniy");
+}
+
C:\Work\lab1.2>
```

Рисунок 5. Последний коммит

2) HEAD~1 : предпоследний коммит.

```
C:\Work\lab1.2>git show HEAD~1
commit 38f749d396e61a6b602957646d6f02542494819a (tag: konec)
Author: Dmitriy <heisenberg3556poul@gmail.com>
Date: Mon May 29 13:38:04 2023 +0300

    6 commit

diff --git a/main.cpp b/main.cpp
index d08eafc..1b0c047 100644
--- a/main.cpp
+++ b/main.cpp
@@ -18,3 +18,11 @@ if (D > 0)
     printf("%lf\n", x1);
     printf("%lf\n", x2);
 }
+else
+{
+    if (D == 0)
+    {
+        x3 = (-b) / (2 * a);
+        printf("%lf\n", x3);
+    }
+}
C:\Work\lab1.2>
```

Рисунок 6. Предпоследний коммит.

3) efe3669: коммит с указанным хэшем.

```
C:\Work\lab1.2>git show efe3669
commit efe3669ae0fd7a303765a0f8ed0b423850591d93
Author: Dmitriy <heisenberg3556poul@gmail.com>
Date: Mon May 29 13:32:35 2023 +0300

    2 commit

diff --git a/main.cpp b/main.cpp
index b40a3a8..be4ef16 100644
--- a/main.cpp
+++ b/main.cpp
@@ -1,3 +1,5 @@
#include <iostream>
#include <conio.h>
#include <math.h>
+int main()
+{
C:\Work\lab1.2>
```

Рисунок 7. Коммит с указанным хэшем.

6. Откат к заданной версии.

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

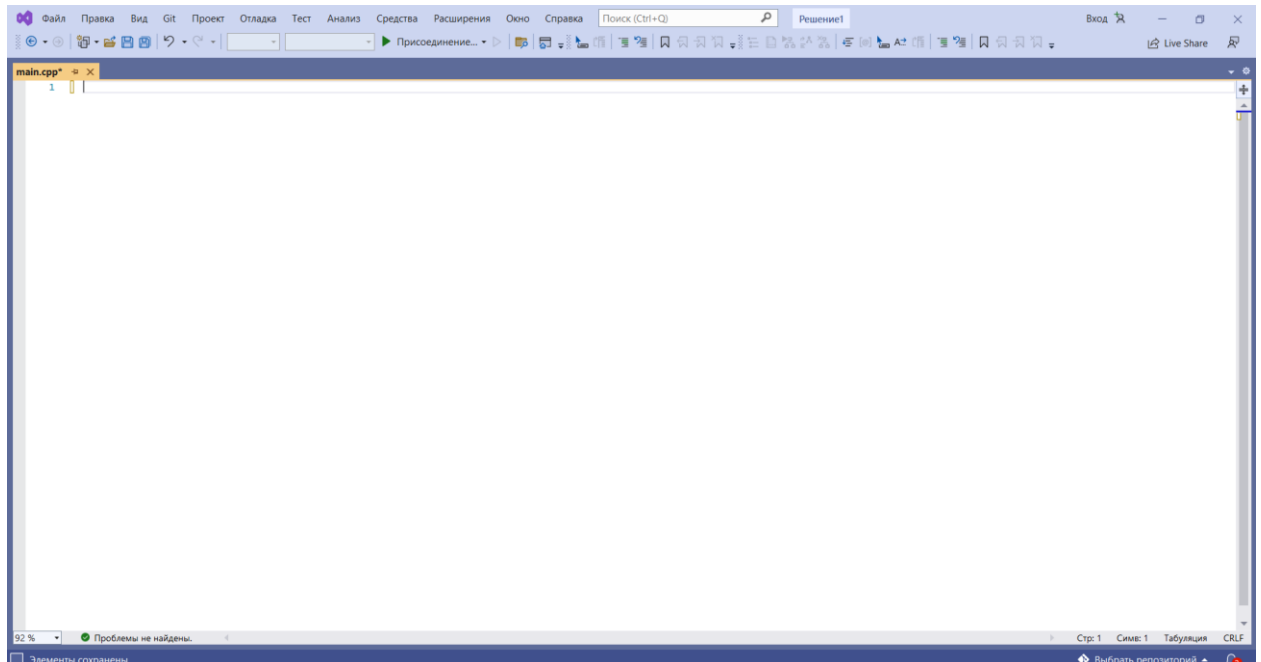


Рисунок 8. Удаление программы

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

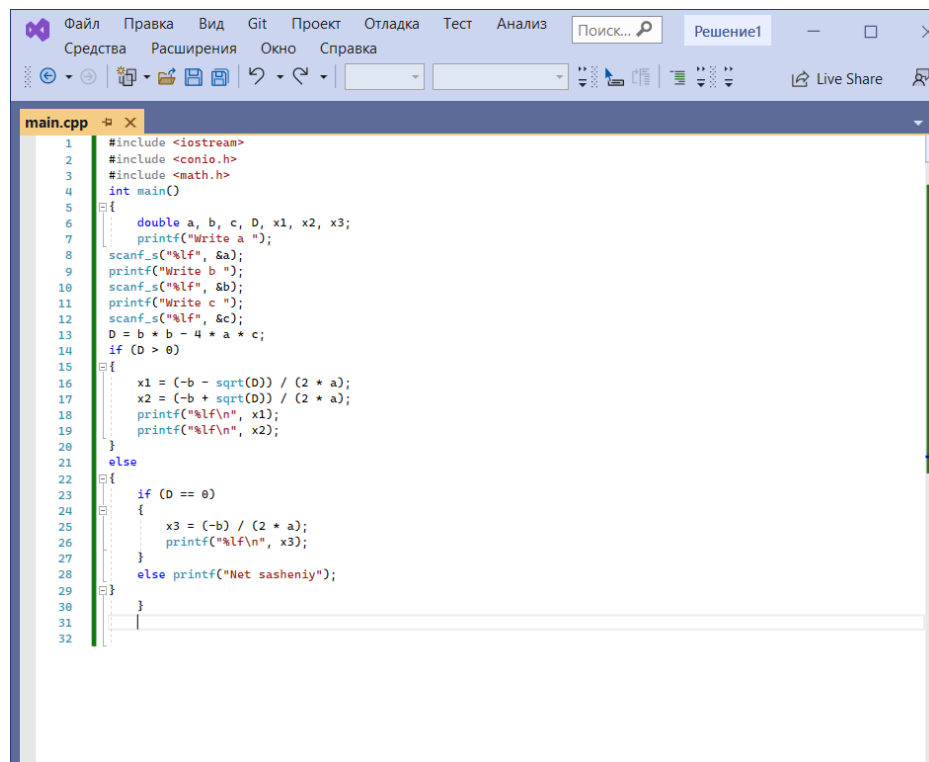
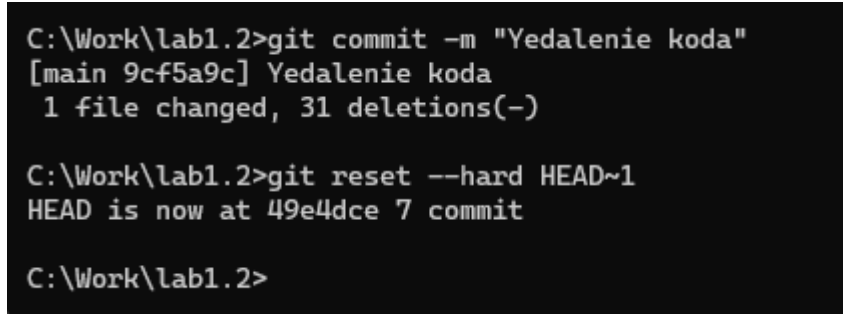


Рисунок 9. Восстановление программы.

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 .



```
C:\Work\lab1.2>git commit -m "Yedalenie koda"
[main 9cf5a9c] Yedalenie koda
1 file changed, 31 deletions(-)

C:\Work\lab1.2>git reset --hard HEAD~1
HEAD is now at 49e4dce 7 commit

C:\Work\lab1.2>
```

Рисунок 10. Возвращение к предпоследней версии коммита

Код вновь вернулся.

Ссылка: <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 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 удалённом компьютере.

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

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