

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ
ФЕДЕРАЦИИ
Федеральное государственное автономное
образовательное учреждение высшего
образования
«СЕВЕРОКАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ
УНИВЕРСИТЕТ»**

**Кафедра
инфокоммуникаций
Институт цифрового
развития**

**ОТЧЁТ
по лабораторной работе №1.2**

Дисциплина: «Основы программной инженерии»

Тема: «Исследование возможностей Git для работы с локальными
репозиториями»

Выполнила:
студентка 2 курса
группы Пиж-б-о-21-1
Джолдошова Мээрим
Бекболотовна

Ставрополь 2022

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

Выполнение работы

1. Был создан общедоступный репозиторий lab_1.2 на GitHub в котором будет использована лицензия MIT и язык программирования C++

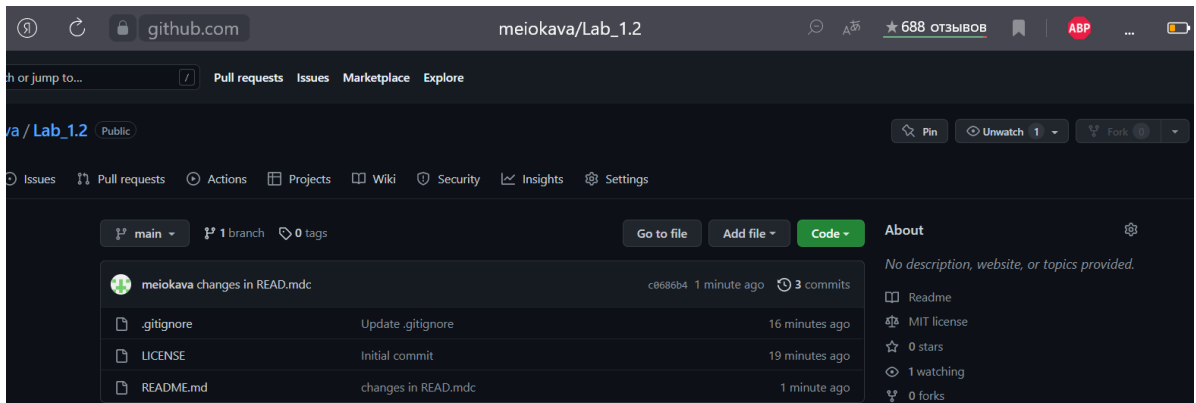


Рисунок 1 – Созданный репозиторий

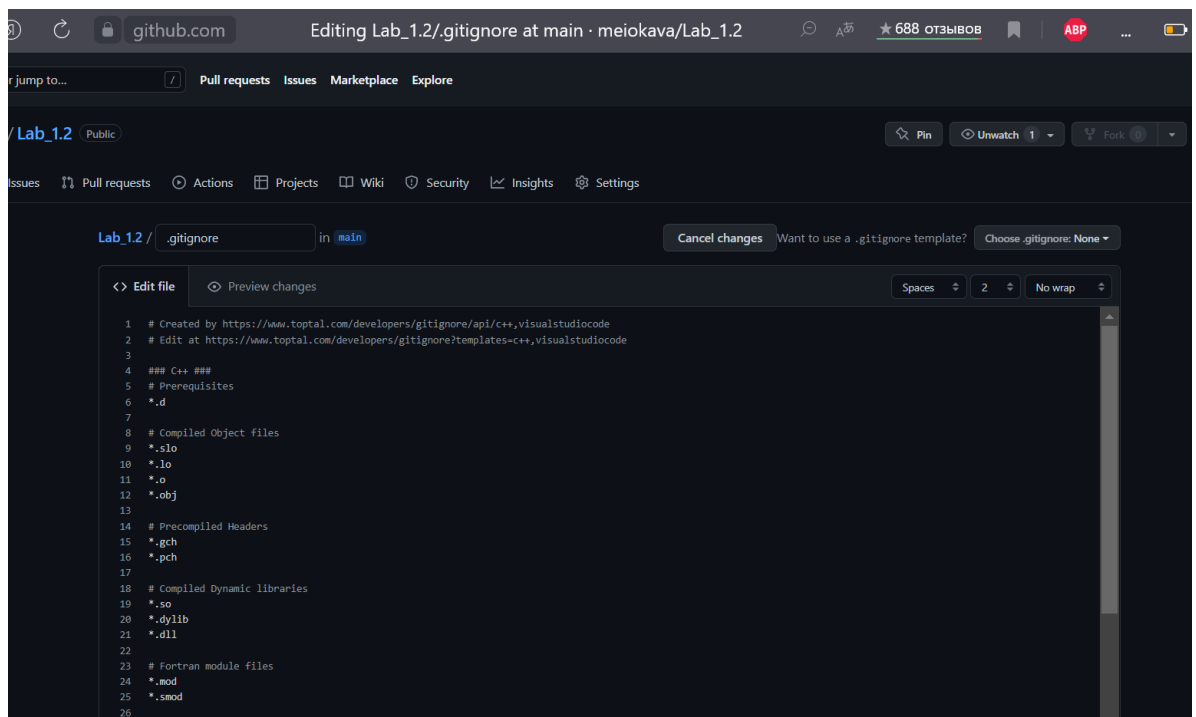


Рисунок 2 – Изменения в файле gitignor для выбранного мной языка программирования

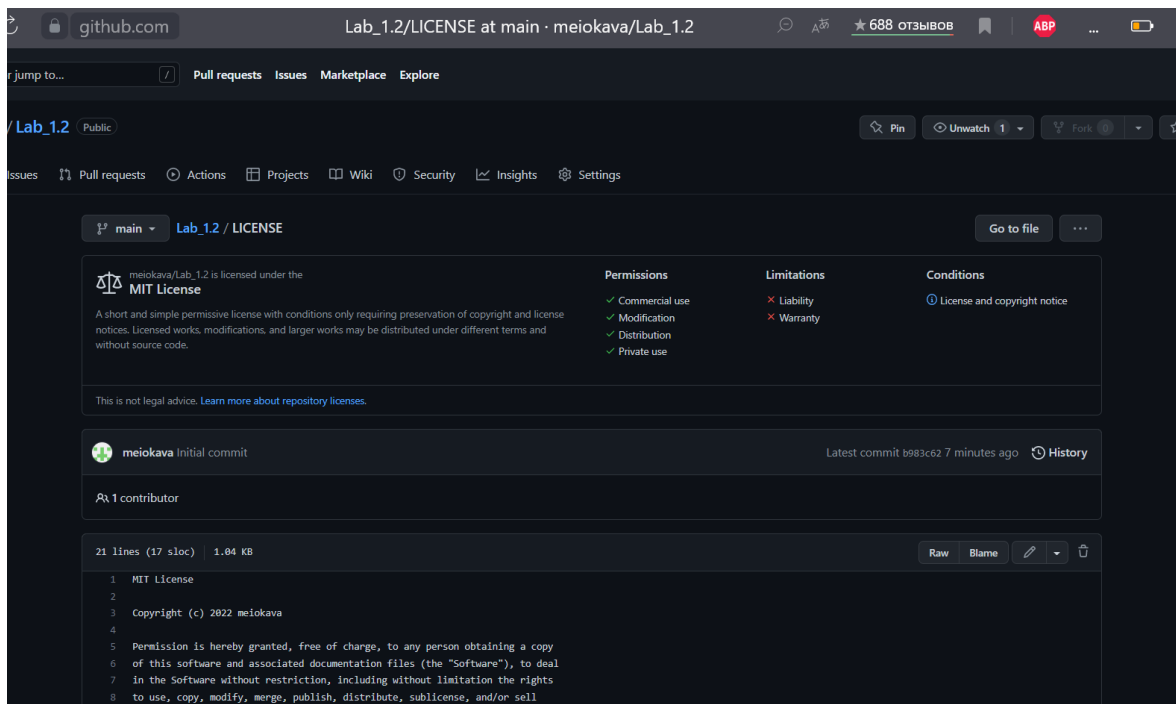


Рисунок 3 – Выбранная лицензия MIT

2. Был клонирован репозиторий на рабочий компьютер

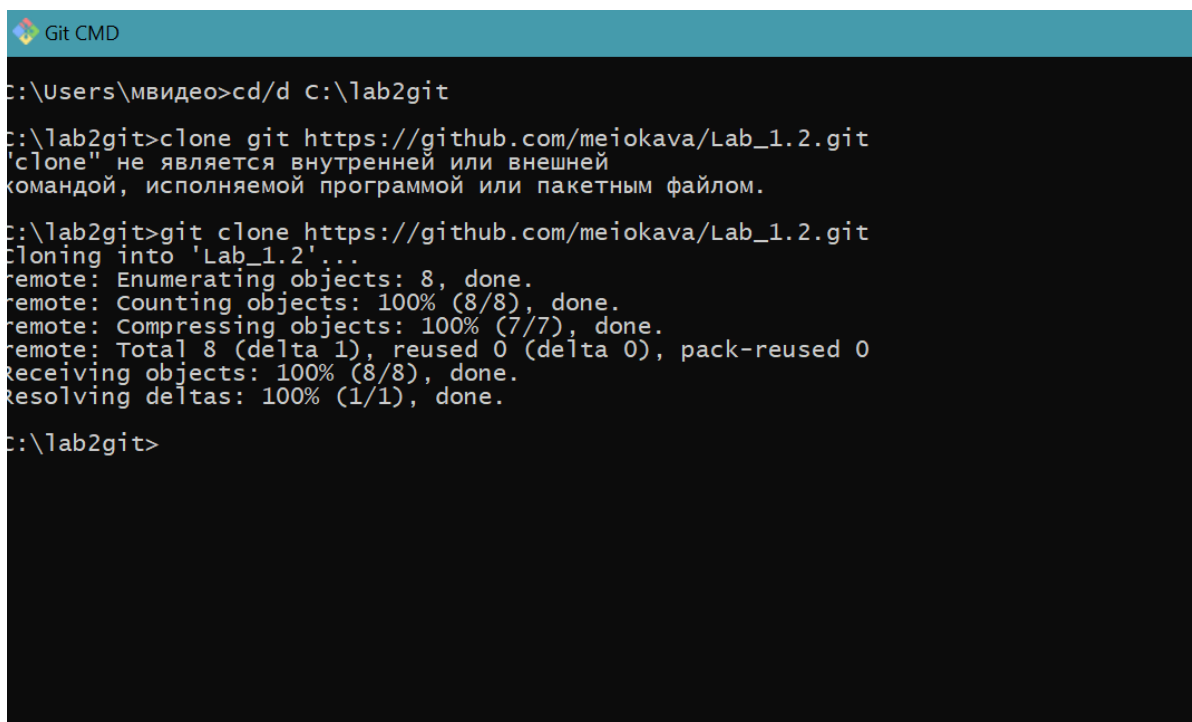


Рисунок 4 – Клонирование репозитория

3. Добавление информации в README

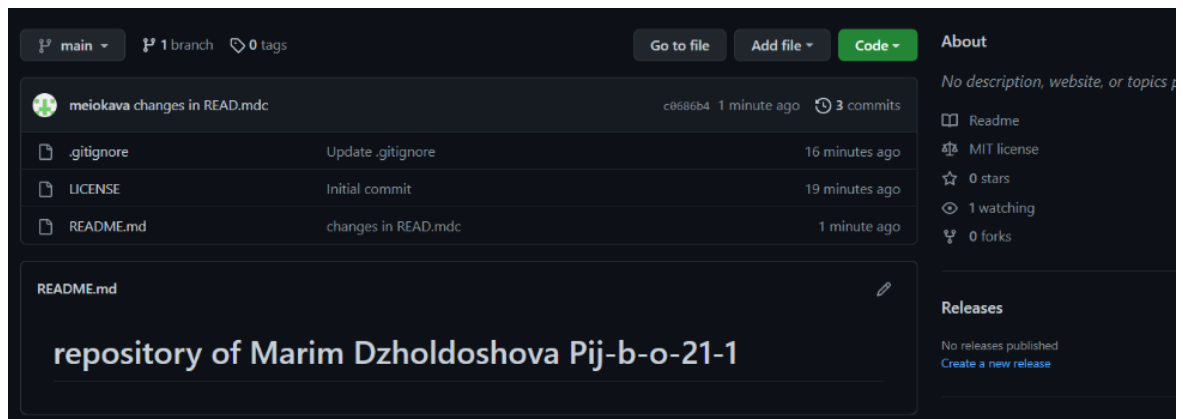


Рисунок 5 – Информация о себе в файле README

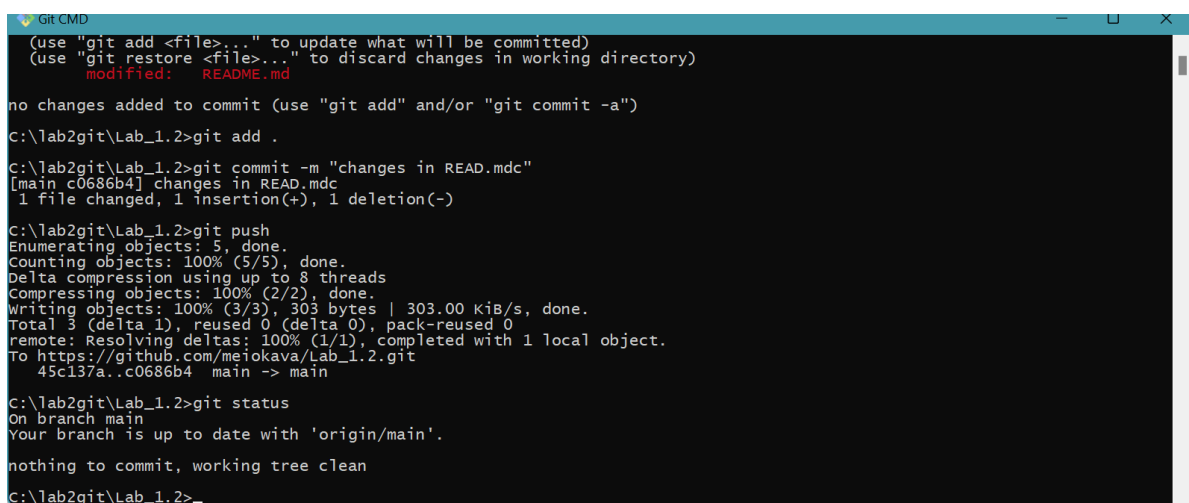


Рисунок 6 – Коммит файла README

4. Было сделано 8 коммитов и добавлено 3 тега

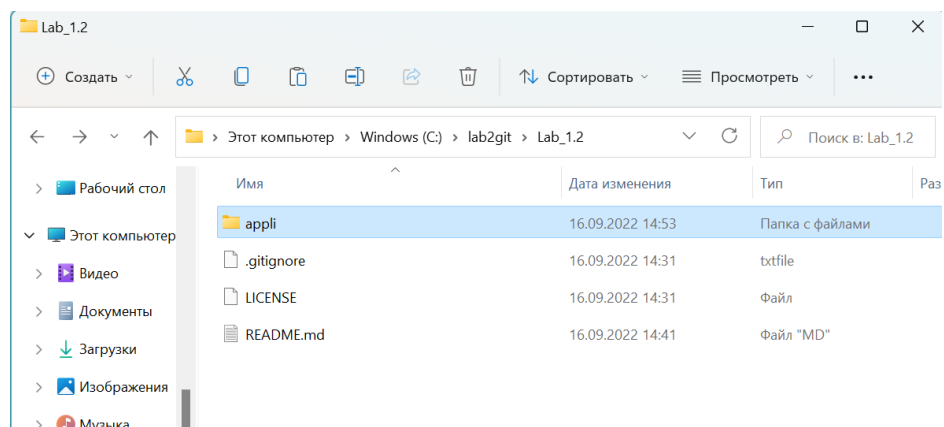


Рисунок 7 – Была создана папка для хранения проекта

```
forGit.cpp*  X
forGit (Глобальная область)
1  #include <iostream>
2  #include <Math.h>
3  using namespace std;
4  int main()
5  {
6      int a = 23;
7      float b = 2.3;
8      float c = 0;
9      c = a + b;
10     cout << "result is"<< c<<endl;
11 }
```

Рисунок 8 – Код программы

```
Git CMD
your branch is up to date with 'origin/main'.
Untracked files:
  (use "git add <file>..." to include in what will be committed)
  appli/

nothing added to commit but untracked files present (use "git add" to track)
c:\lab2git\Lab_1.2>git add .
c:\lab2git\Lab_1.2>git commit -m "new application"
[main 151fdd4] new application
4 files changed, 210 insertions(+)
create mode 100644 appli/forGit.sln
create mode 100644 appli/forGit/forGit.cpp
create mode 100644 appli/forGit/forGit.vcxproj
create mode 100644 appli/forGit/forGit.vcxproj.filters
c:\lab2git\Lab_1.2>git push
Enumerating objects: 9, done.
Counting objects: 100% (9/9), done.
Delta compression using up to 8 threads
compressing objects: 100% (8/8), done.
writing objects: 100% (8/8), 2.59 KiB | 883.00 KiB/s, done.
Total 8 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/meiokava/Lab_1.2.git
   c0686b4..151fdd4  main -> main
c:\lab2git\Lab_1.2>
```

Рисунок 9 – Коммит и пуш программы на удаленный сервис



Рисунок 10 – Изменения на удаленном сервере

```
Git CMD
c:\lab2git\Lab_1.2>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:   appli/forGit/forGit.cpp

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

c:\lab2git\Lab_1.2>git add .
c:\lab2git\Lab_1.2>git commit -m "eighth changes"
[main 0fc015b] eighth changes
1 file changed, 1 insertion(+)

c:\lab2git\Lab_1.2>git push
Enumerating objects: 9, done.
Counting objects: 100% (9/9), done.
Delta compression using up to 8 threads
Compressing objects: 100% (5/5), done.
Writing objects: 100% (5/5), 470 bytes | 156.00 KiB/s, done.
Total 5 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To https://github.com/meiokava/Lab_1.2.git
   eab3dd8..0fc015b  main -> main

c:\lab2git\Lab_1.2>git tag a- version 3.5.1 -m
error: switch 'm' requires a value
```

Рисунок 11 – Было сделано 8 коммитов

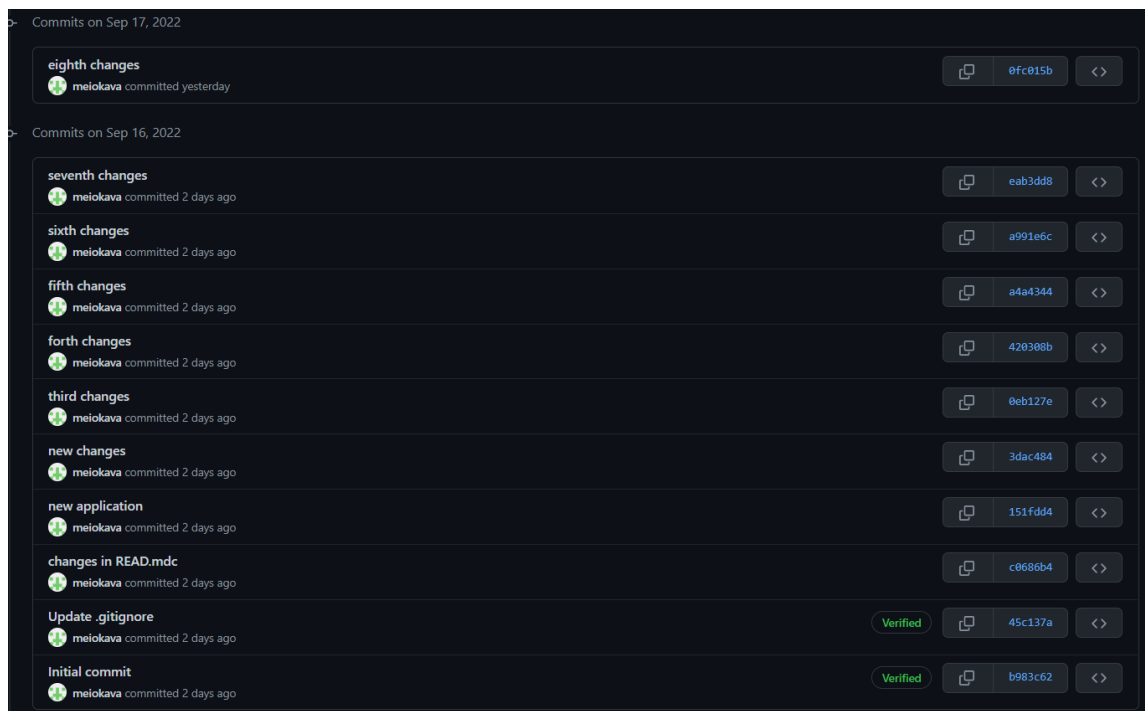


Рисунок 12 – История коммитов на удаленном сервере

```
Git CMD
c:\lab2git\Lab_1.2>git tag -a ver-2.5 -m "beta version 2.5"
c:\lab2git\Lab_1.2>git tag
ver-2.5
c:\lab2git\Lab_1.2>git show version-2.5
fatal: ambiguous argument 'version-2.5': unknown revision or path not in the working tree.
Use '--' to separate paths from revisions, like this:
'git <command> [<revision>...] -- [<file>...]'
c:\lab2git\Lab_1.2>git show
commit a991e6c5e4948e75db72ee63c6eef3d89212d1ba (HEAD -> main, tag: ver-2.5, origin/main, origin/HEAD)
Author: meikava <meikava7@gmail.com>
Date:   Fri Sep 16 16:24:29 2022 +0300

    sixth changes

diff --git a/appli/forGit/forGit.cpp b/appli/forGit/forGit.cpp
index ca6b25d..dcb8fa5 100644
--- a/appli/forGit/forGit.cpp
+++ b/appli/forGit/forGit.cpp
@@ -8,7 +8,8 @@ int main()
     cout << "result is" << endl;
     cout << a + b << endl;
     cout << "it is a result";
-    cout << 10 + 1<<endl;
+    cout << 10 + 3<<endl;
+    cout << 10 + 1 << endl;
```

Рисунок 13 – Создание аннотированного тега

```
C:\lab2git\Lab_1.2>git tag version-3.0
C:\lab2git\Lab_1.2>git n
git: 'n' is not a git command. See 'git --help'.

The most similar command is
    svn
C:\lab2git\Lab_1.2>git tag -n
ver-2.5      beta version 2.5
version-3.0  seventh changes
C:\lab2git\Lab_1.2>
```

Рисунок 14 – Создание легковесного тега

```
C:\lab2git\Lab_1.2>git tag -n
ver-2.5      beta version 2.5
version-3.0  seventh changes

C:\lab2git\Lab_1.2>git tag -a version-3.5 -m "beta version 3.5"

C:\lab2git\Lab_1.2>git tag -n
ver-2.5      beta version 2.5
version-3.0  seventh changes
version-3.5  beta version 3.5
```

Рисунок 15 – Создание третьего тега

```
Git CMD
version-3.5      beta version 3.5
1.
C:\lab2git\Lab_1.2>git tag -a ver 2.0 -m "fifth changes"
fatal: Failed to resolve '2.0' as a valid ref.

C:\lab2git\Lab_1.2>git tag -a version 4.0 -m "fifth changes"
fatal: Failed to resolve '4.0' as a valid ref.

C:\lab2git\Lab_1.2>git push origin --tags
Enumerating objects: 2, done.
Counting objects: 100% (2/2), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (2/2), 288 bytes | 288.00 KiB/s, done.
Total 2 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/meikava/Lab_1.2.git
 * [new tag]         ver-2.5 -> ver-2.5
 * [new tag]         version-3.0 -> version-3.0
 * [new tag]         version-3.5 -> version-3.5

C:\lab2git\Lab_1.2>
```

Рисунок 16 – Отправка тегов на удаленный сервер

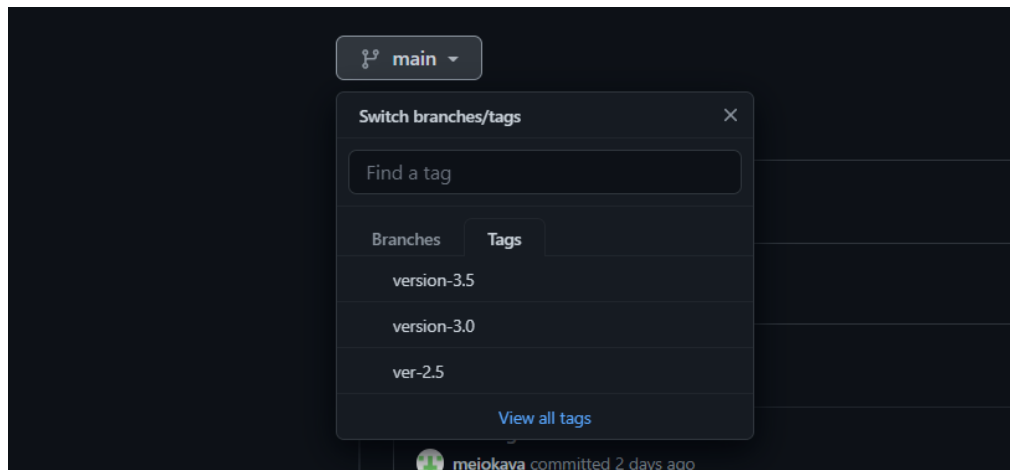


Рисунок 17 – История тегов на удаленном сервере

```
c:\lab2git\Lab_1.2>git log
commit 0fc015bed4760278984416cfeb1576f0c37614ae (HEAD -> main, tag: version-3.5, origin/main, origin/HEAD)
Author: meiokava <meiokava7@gmail.com>
Date: Sat Sep 17 11:18:20 2022 +0300

eighth changes

commit eab3dd89ecfe6727d4c0bd3843ed092e41ef9445 (tag: version-3.0)
Author: meiokava <meiokava7@gmail.com>
Date: Fri Sep 16 16:38:05 2022 +0300

seventh changes

commit a991e6c5e4948e75db72ee63c6eef3d89212d1ba (tag: ver-2.5)
Author: meiokava <meiokava7@gmail.com>
Date: Fri Sep 16 16:24:29 2022 +0300

sixth changes

commit a4a4344801c561ebae6928e677cef257449dd10
Author: meiokava <meiokava7@gmail.com>
Date: Fri Sep 16 16:23:15 2022 +0300

fifth changes

commit 420308b3aaf9cef8093db787d5e73305b3e9205c
Author: meiokava <meiokava7@gmail.com>
Date: Fri Sep 16 16:21:59 2022 +0300

forth changes
...skipping...
commit 0fc015bed4760278984416cfeb1576f0c37614ae (HEAD -> main, tag: version-3.5, origin/main, origin/HEAD)
Author: meiokava <meiokava7@gmail.com>
Date: Sat Sep 17 11:18:20 2022 +0300

eighth changes

commit eab3dd89ecfe6727d4c0bd3843ed092e41ef9445 (tag: version-3.0)
Author: meiokava <meiokava7@gmail.com>
```

Рисунок 18 – История коммитов при помощи команды git log


```

commit 0fc015bed4760278984416cfb1576f0c37614ae (HEAD -> main, tag: version-3.5, origin/main, origin/HEAD)
author: meikava <meikava7@gmail.com>
date: Sat Sep 17 11:18:20 2022 +0300

    eighth changes

commit eab3dd89ecfe6727d4c0bd3843ed092e41ef9445 (tag: version-3.0)
author: meikava <meikava7@gmail.com>
date: Fri Sep 16 16:38:05 2022 +0300

    seventh changes

commit a991e6c5e4948e75db72ee63c6eef3d89212d1ba (tag: ver-2.5)
author: meikava <meikava7@gmail.com>
date: Fri Sep 16 16:24:29 2022 +0300

    sixth changes

commit a4a4344801c561ebae6928e677cef257449dd10
author: meikava <meikava7@gmail.com>
date: Fri Sep 16 16:23:15 2022 +0300

    fifth changes

commit 420308b3aaf9cef8093db787d5e73305b3e9205c
author: meikava <meikava7@gmail.com>
date: Fri Sep 16 16:21:59 2022 +0300

    forth changes

commit 0eb127e92ec7e466911542eac743f8c8a8e9e4b0
author: meikava <meikava7@gmail.com>
date: Fri Sep 16 16:17:55 2022 +0300

    third changes

commit 3dac484f0120b5d5844dd967b77713abc9dfd2db
author: meikava <meikava7@gmail.com>
date: Fri Sep 16 16:16:10 2022 +0300

    new changes

commit 151fdd4bbaff81ccbfb9d262cc6f9757084d703
author: meikava <meikava7@gmail.com>
date: Fri Sep 16 14:53:56 2022 +0300

```

Рисунок 19 – Продолжение истории коммитов

```

C:\lab2git\Lab_1.2>git log --graph --pretty=oneline --abbrev-commit
* 0fc015b (HEAD -> main, tag: version-3.5, origin/main, origin/HEAD) eighth changes
* eab3dd8 (tag: version-3.0) seventh changes
* a991e6c (tag: ver-2.5) sixth changes
* a4a4344 fifth changes
* 420308b forth changes
* 0eb127e third changes
* 3dac484 new changes
* 151fdd4 new application
* c0686b4 changes in READ.mdc
* 45c137a Update .gitignore
* b983c62 Initial commit

C:\lab2git\Lab_1.2>_

```

Рисунок 20 – Просмотр коммитов командой git graph

5. Просмотрел содержимое коммитов командой git show HEAD, git show HEAD~1, git show a991e6c:

```

c:\lab2git\Lab_1.2>git show head
commit 0fc015bed4760278984416cfef1576f0c37614ae (HEAD -> main, tag: version-3.5, origin/main, origin/HEAD)
Author: meikava <meikava7@gmail.com>
Date:   Sat Sep 17 11:18:20 2022 +0300

    eighth changes

diff --git a/appli/forGit/forGit.cpp b/appli/forGit/forGit.cpp
index 32fd72b..84216cf 100644
--- a/appli/forGit/forGit.cpp
+++ b/appli/forGit/forGit.cpp
@@ -13,4 +13,5 @@ int main()
     cout << "it is a result";
     cout << "it is a result";
     cout << "it is rainy day";
+    cout << "it is rainy day";
 }
\ No newline at end of file
c:\lab2git\Lab_1.2>_

```

Рисунок 21 – Просмотр содержимого последнего коммита

```

c:\lab2git\Lab_1.2>git show head~1
commit eab3dd89ecfe6727d4c0bd3843ed092e41ef9445 (tag: version-3.0)
Author: meikava <meikava7@gmail.com>
Date:   Fri Sep 16 16:38:05 2022 +0300

    seventh changes

diff --git a/appli/forGit/forGit.cpp b/appli/forGit/forGit.cpp
index dcb8fa5..32fd72b 100644
--- a/appli/forGit/forGit.cpp
+++ b/appli/forGit/forGit.cpp
@@ -12,4 +12,5 @@ int main()
     cout << 21 + 3 << endl;
     cout << "it is a result";
     cout << "it is a result";
+    cout << "it is rainy day";
 }
\ No newline at end of file
c:\lab2git\Lab_1.2>_

```

Рисунок 22 – Просмотр предпоследнего коммита

```

c:\lab2git\Lab_1.2>git show a991e6c
commit a991e6c5e4948e75db72ee63c6eef3d89212d1ba (tag: ver-2.5)
Author: meikava <meikava7@gmail.com>
Date:   Fri Sep 16 16:24:29 2022 +0300

    sixth changes

diff --git a/appli/forGit/forGit.cpp b/appli/forGit/forGit.cpp
index ca6b25d..dcb8fa5 100644
--- a/appli/forGit/forGit.cpp
+++ b/appli/forGit/forGit.cpp
@@ -8,7 +8,8 @@ int main()
     cout << "result is" << endl;
     cout << a + b << endl;
     cout << "it is a result";
-    cout << 10 + 1 << endl;
-    cout << 21 + 3 << endl;
+    cout << 10 + 1 << endl;
+    cout << 21 + 3 << endl;
+    cout << "it is a result";
     cout << "it is a result";
 }
\ No newline at end of file
c:\lab2git\Lab_1.2>

```

Рисунок 23 – Просмотр коммита с указанным хэшем

6. Возможность отката к заданной версии

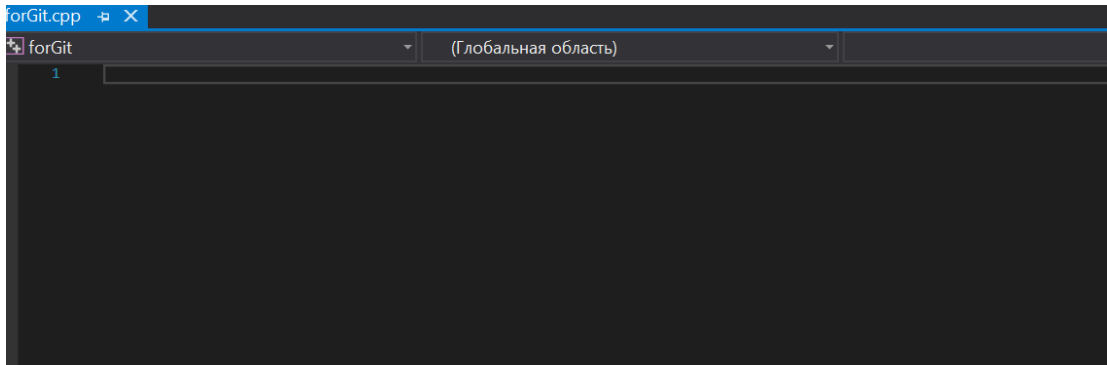


Рисунок 24 – Весь код был удален, а изменения сохранены

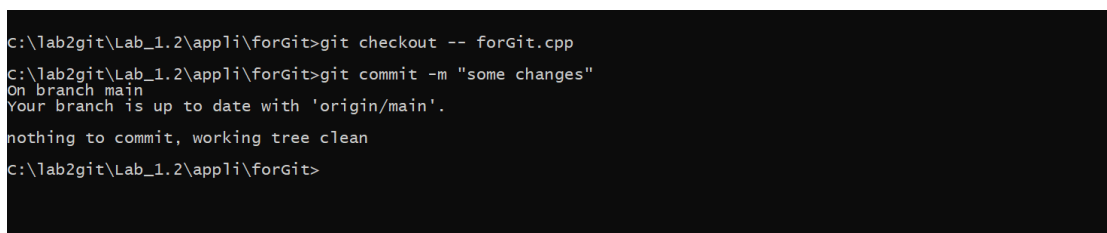


Рисунок 25 – Удаление всех несохраненных изменений командой checkout

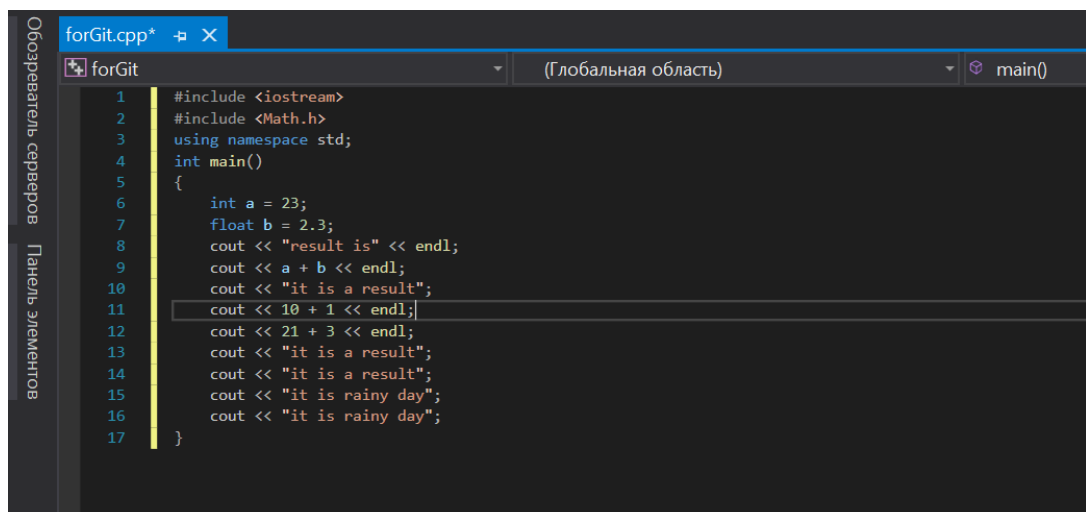


Рисунок 26 – Изменения программы после команды checkout

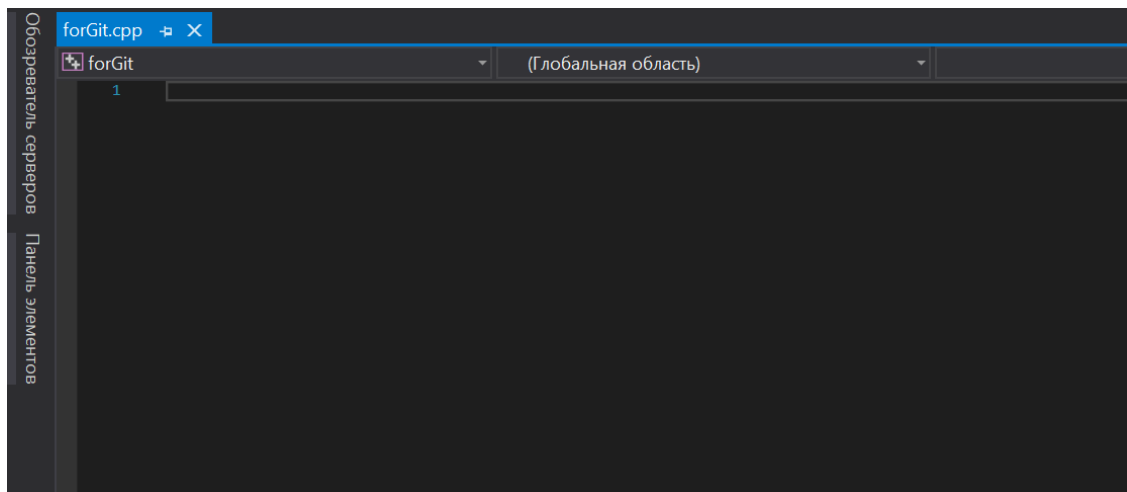


Рисунок 27 – Удаление кода в Visual studio

```
c:\lab2git\Lab_1.2\appli\forGit>git checkout -- forGit.cpp
c:\lab2git\Lab_1.2\appli\forGit>git commit -m "some changes"
On branch main
Your branch is up to date with 'origin/main'.
nothing to commit, working tree clean
c:\lab2git\Lab_1.2\appli\forGit>git commit -m "delete all garbage"
```

Рисунок 28 – Коммит изменений

```
c:\lab2git\Lab_1.2\appli\forGit>git reset --hard head~1
HEAD is now at eab3dd8 seventh changes
c:\lab2git\Lab_1.2\appli\forGit>_
```

Рисунок 29 – Откат состояния хранилища к предыдущей версии

Вывод: в ходе работы были изучены базовые возможности системы контроля версий Git для работы с локальными репозиториями

Контрольные вопросы и ответы на них:

Вопросы для защиты работы.

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

Наиболее простой и в то же время мощный инструмент для этого — команда `git log`. По умолчанию, без аргументов, `git log` выводит список

коммитов созданных в данном репозитории в обратном хронологическом порядке. То есть самые последние коммиты показываются первыми. Одна из опций, когда вы хотите увидеть сокращенную статистику для каждого коммита, вы можете использовать опцию `--stat`.

Вторая опция (одна из самых полезных аргументов) является `-p` или `--patch`, который показывает разницу (выводит патч), внесенную в каждый коммит. Так же вы можете ограничить количество записей в выводе команды; используйте параметр `-2` для вывода только двух записей (пример команды `git log -p -2`).

Третья действительно полезная опция это `--pretty`. Она меняет формат вывода. Существует несколько встроенных вариантов отображения. Опция `oneline` выводит каждый коммит в одну строку, что может быть очень удобным если вы просматриваете большое количество коммитов. К тому же, опции `short`, `full` и `fuller` делают вывод приблизительно в том же формате, но с меньшим или большим количеством информации соответственно. Наиболее интересной опцией является `format`, которая позволяет указать формат для вывода информации. Особенно это может быть полезным, когда вы хотите сгенерировать вывод для автоматического анализа — так как вы указываете формат явно, он не будет изменен даже после обновления Git. Для опции `git log --pretty=format` существуют различного рода опции для изменения формата отображения.

2. Как ограничить вывод при просмотре истории коммитов?

Для ограничения может использоваться функция `git log <n>`, где `n` число записей. Также, существуют опции для ограничения вывода по времени, такие как `--since` и `--until`, они являются очень удобными. Например, следующая команда покажет список коммитов, сделанных за последние две недели: `git log --since=2. weeks` Это команда работает с большим количеством форматов — вы можете указать определенную дату вида `2008-01-15` или же относительную дату, например `2 years 1 day 3 minutes ago`.

Также вы можете фильтровать список коммитов по заданным

параметрам. Опция `--author` дает возможность фильтровать по автору коммита, а опция `--grep` (показывает только коммиты, сообщение которых содержит указанную строку) искать по ключевым словам в сообщении коммита.

Функция `-S` показывает только коммиты, в которых изменение в коде повлекло за собой добавление или удаление указанной строки.

3. Как внести изменения в уже сделанный коммит?

Внести изменения можно с помощью команды `git commit --amend`. Эта команда берёт индекс и применяет его к последнему коммиту. Если после последнего коммита не было никаких проиндексированных изменений (например, вы запустили приведённую команду сразу после предыдущего коммита), то состояние проекта будет абсолютно таким же и всё, что мы изменим, это комментарий к коммиту.

Для того, чтобы внести необходимые изменения - нам нужно проиндексировать их и выполнить команду `git commit --amend`.
`git commit -m 'initial commit' git add forgotten_file git commit --amend`

Эффект от выполнения этой команды такой, как будто мы не выполнили предыдущий коммит, а еще раз выполнили команду `git add` и выполнили коммит.

4. Как отменить индексацию файла в Git?

Например, вы изменили два файла и хотите добавить их в разные коммиты, но случайно выполнили команду `git add *` и добавили в индекс оба. Как исключить из индекса один из них? Команда `git status` напомним вам: прямо под текстом «Changes to be committed» говорится: используйте `git reset HEAD <file>` для исключения из индекса.

5. Как отменить изменения в файле? С помощью команды `git checkout - - <file>`.

6. Что такое удаленный репозиторий Git? Удалённый репозиторий — это своего рода наше облако, в которое мы сохраняем те или иные изменения в нашей программе/коде/файлах.

7. Как выполнить просмотр удаленных репозиториях данного

локального репозитория?

Для того, чтобы просмотреть список настроенных удалённых репозиторий, необходимо запустить команду `git remote`. Также можно указать ключ `-v`, чтобы просмотреть адреса для чтения и записи, привязанные к репозиторию. Пример: `git remote -v`

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?

Если необходимо получить изменения, которые есть у Пола, но нету у вас, вы можете выполнить команду `git fetch <Название репозитория>`. Важно отметить, что команда `git fetch` забирает данные в ваш локальный репозиторий, но не сливает их с какими-либо вашими наработками и не модифицирует то, над чем вы работаете в данный момент. Вам необходимо вручную слить эти данные с вашими, когда вы будете готовы. Если ветка настроена на отслеживание удалённой ветки, то вы можете использовать команду `git pull` чтобы автоматически получить изменения из удалённой ветки и слить их со своей текущей. Выполнение `git pull`, как правило, извлекает (`fetch`) данные с сервера, с которого вы изначально клонировали, и автоматически пытается слить (`merge`) их с кодом, над которым вы в данный момент работаете. Чтобы отправить изменения на удалённый репозиторий необходимо отправить их в удалённый репозиторий. Команда для этого действия простая: `git push <remote-name> <branch-name>`.

10. Как выполнить просмотр удаленного репозитория?

Для просмотра удалённого репозитория, можно использовать команду `git remote show <remote>`.

11. Каково назначение тэгов Git?

Теги — это ссылки, указывающие на определённые версии

кода/написанной программы. Они удобны чтобы в случае чего вернуться к нужному моменту. Также при помощи тегов можно помечать важные моменты.

12. Как осуществляется работа с тэгами Git?

Просмотреть наличие тегов можно с помощью команды: `git tag`. А назначить (указать, добавить тег) можно с помощью команды `git tag -a v1.4(версия изначальная) -m "Название"`. С помощью команды `git show` вы можете посмотреть данные тега вместе с коммитом: `git show v1.4`. Отправка тегов, 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 отправки всех тегов можно использовать команду `git push origin tags`. Для удаления тега в локальной репозитории достаточно выполнить команду `git tag -d <tagname>`. Например, удалить созданный ранее легковесный тег можно следующим образом: `git tag -d v1.4-lw`. Для удаления тега из внешнего репозитория используется команда `git push origin --delete <tagname>`. Если вы хотите получить версии файлов, на которые указывает тег, то вы можете сделать `git checkout` для тега пример: `git checkout -b version2 v2.0.0`.

13. Самостоятельно изучите назначение флага --prune в командах git fetch и git push. Каково назначение этого флага? `Git fetch --prune` команда получения всех изменений с репозитория GitHub. В команде `git push --prune` удаляет удаленные ветки, у которых нет локального аналога.