

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

Кафедра инфокоммуникаций

**«Исследование возможностей Git для работы с локальными
репозиториями»**

**Отчет по лабораторной работе № 1.2
по дисциплине «Основы кроссплатформенного
программирования»**

Выполнил студент группы ИВТ-б-о-21-1

Шайдеров Дмитрий Викторович.

«13 » апреля 2022г.

Подпись студента _____

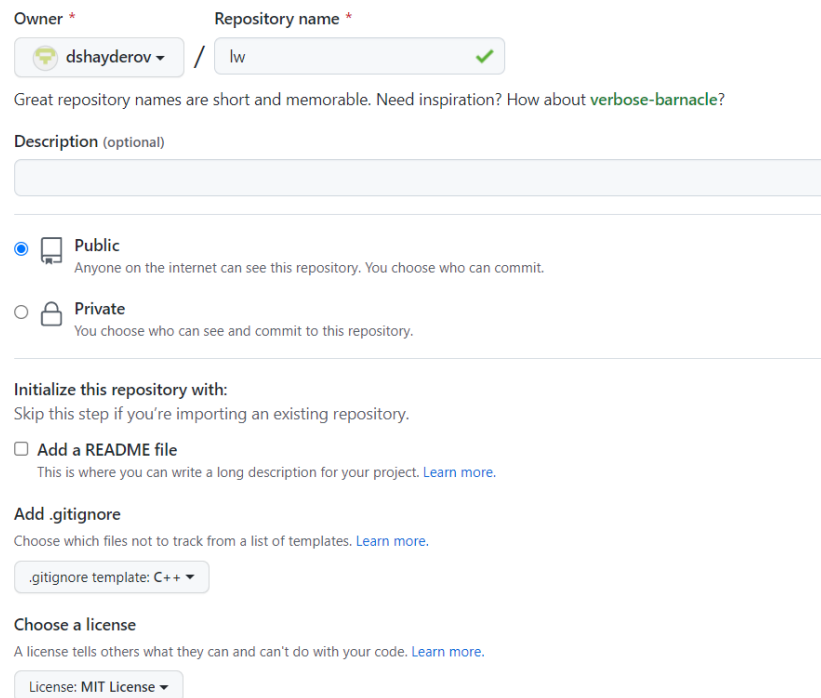
Работа защищена « » _____ 20__ г.

Проверил Воронкин Р.А. _____
(подпись)

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

Порядок выполнения работы:

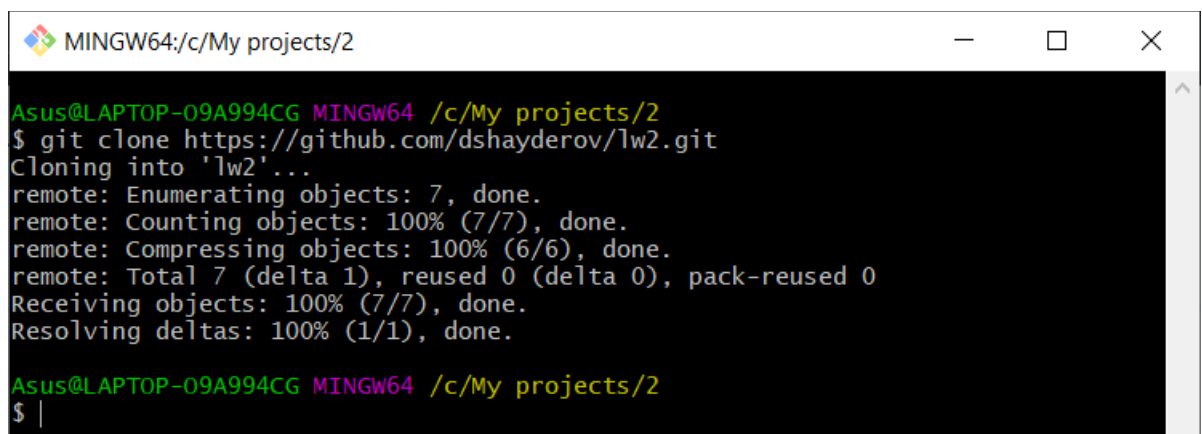
1. Создал общедоступный репозиторий на GitHub, в котором использована лицензия MIT и язык программирования C++.



The screenshot shows the GitHub 'Create new repository' form. The 'Owner' is 'dshayderov' and the 'Repository name' is 'lw', which is marked with a green checkmark. Below the name, there is a suggestion: 'Great repository names are short and memorable. Need inspiration? How about [verbose-barnacle](#)?'. The 'Description' field is empty. Under 'Visibility', the 'Public' option is selected with a radio button, and a description says 'Anyone on the internet can see this repository. You choose who can commit.' The 'Private' option is unselected. The 'Initialize this repository with:' section has a checkbox for 'Add a README file' which is unchecked. Below it, there is a section for 'Add .gitignore' with a dropdown menu set to '.gitignore template: C++'. At the bottom, there is a 'Choose a license' section with a dropdown menu set to 'License: MIT License'.

Рисунок 1 - Создание репозитория

2. Выполните клонирование созданного репозитория на рабочий компьютер.



```
MINGW64:/c/My projects/2
Asus@LAPTOP-09A994CG MINGW64 /c/My projects/2
$ git clone https://github.com/dshayderov/lw2.git
Cloning into 'lw2'...
remote: Enumerating objects: 7, done.
remote: Counting objects: 100% (7/7), done.
remote: Compressing objects: 100% (6/6), done.
remote: Total 7 (delta 1), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (7/7), done.
Resolving deltas: 100% (1/1), done.
Asus@LAPTOP-09A994CG MINGW64 /c/My projects/2
$ |
```

Рисунок 2 - Клонирование репозитория

3. Дополнил файл .gitignore

```

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 Средства разработки на языках C++, C и ассемблера, а также библиотеки доступны в составе Visual Studio в Windows.
35 Можно использовать C++ в Visual Studio для создания любых решений:
36 простые консольные приложения
37 классические приложений для Windows
38 драйвера устройств и компонентов операционных систем
39 кроссплатформатные игры для мобильных устройств
40 системы для небольших устройств Интернета вещей
41 многосерверные вычислительные платформы в облаке Azure

```

Рисунок 3 - Файл .gitignore

4. Добавил в файл README.md информацию о дисциплине, группе и ФИО студента, выполняющего лабораторную работу.

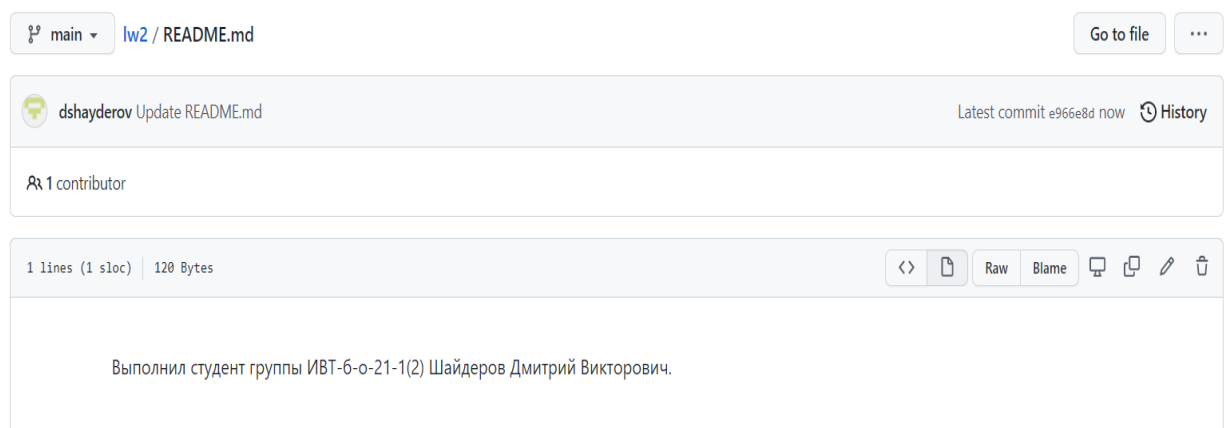


Рисунок 4 - Файл README.md

5. Напишите программу. Фиксировал изменения при написании программы в локальном репозитории. Сделано 7 коммитов, отмеченных 3 тэгами.

```

1  #include <iostream>
2  #include <cmath>
3  using namespace std;
4
5  int main()
6  {
7      cout << "Calculator\n";
8      int s1, s2, sub1, sub2, mul1, mul2, div1, div2, ch;
9      cout << "Choose the action: \n1-Addition\n2-Subtraction\n3-Multiplication\n4-Division\n";
10     cin >> ch;
11     if (ch == 1)
12     {
13         cout << "Addition\n" << "Enter s1 and s2:";
14         cin >> s1 >> s2;
15         cout << "Result: " << s1 + s2 << endl;
16     }
17     if (ch == 2)
18     {
19         cout << "Subtraction\n" << "Enter sub1 and sub2: ";
20         cin >> sub1 >> sub2;
21         cout << "Result: " << sub1 - sub2 << endl;
22     }
23     if (ch == 3)
24     {
25         cout << "Multiplication\n" << "Enter mul1 and mul2: ";
26         cin >> mul1 >> mul2;
27         cout << "Result: " << mul1 * mul2 << endl;
28     }
29     if (ch == 4)
30     {
31         cout << "Division\n" << "Enter div1 and div2: ";
32         cin >> div1 >> div2;
33         cout << "Result: " << div1 / div2 << endl;
34     }
35     return 0;
36 }

```

Рисунок 5 - Конечный вид программы

Упрощение программы dshayderov committed 10 minutes ago	81489f2	<>
Добавлен выбор действия dshayderov committed 13 minutes ago	4b28a85	<>
Добавлено деление dshayderov committed 20 minutes ago	1b740b4	<>
Добавлено умножение dshayderov committed 24 minutes ago	54a0cb9	<>
Добавлено вычитание dshayderov committed 28 minutes ago	9b6c9f6	<>
Добавлено сложение dshayderov committed 38 minutes ago	bbc0cf5	<>
Новый проект dshayderov committed 1 hour ago	16b069d	<>
Update README.md dshayderov committed 1 hour ago	Verified e966e8d	<>
Update .gitignore dshayderov committed 1 hour ago	Verified ae5350d	<>
Update .gitignore dshayderov committed 2 hours ago	Verified 7c16d79	<>
Create README.md dshayderov committed 4 hours ago	Verified 8c6962e	<>
Initial commit dshayderov committed 4 hours ago	Verified 6881c72	<>

Рисунок 6 – Коммиты

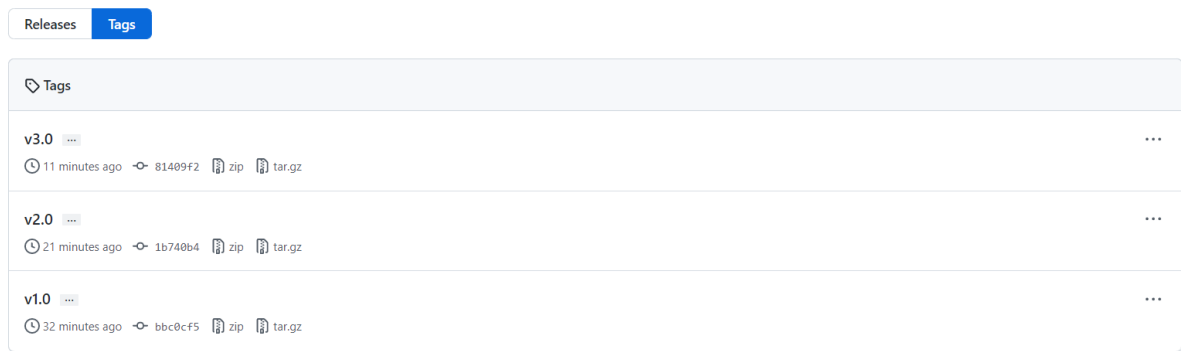


Рисунок 7 - Теги

6. Просмотрел историю (журнал) хранилища командой `git log --graph --pretty=oneline --abbrev-commit`.

```
MINGW64:/c/My projects/2/lw2

Asus@LAPTOP-09A994CG MINGW64 /c/My projects/2/lw2 (main)
$ git log --graph --pretty=oneline --abbrev-commit
* a16dc9a (HEAD -> main, origin/main, origin/HEAD) Merge branch 'main' of http
s://github.com/dshayderov/lw2
|
| * e966e8d Update README.md
| * ae5350d Update .gitignore
| * 7c16d79 Update .gitignore
| * 81409f2 (tag: v3.0) Упрощение программы
| * 4b28a05 Добавлен выбор действия
| * 1b740b4 (tag: v2.0) Добавлено деление
| * 54a0cb9 Добавлено умножение
| * 9b6c9f6 Добавлено вычитание
| * bbc0cf5 (tag: v1.0) Добавлено сложение
| * 16b069d Новый проект
|/
* 8c6962e Create README.md
* 6881c72 Initial commit

Asus@LAPTOP-09A994CG MINGW64 /c/My projects/2/lw2 (main)
$
```

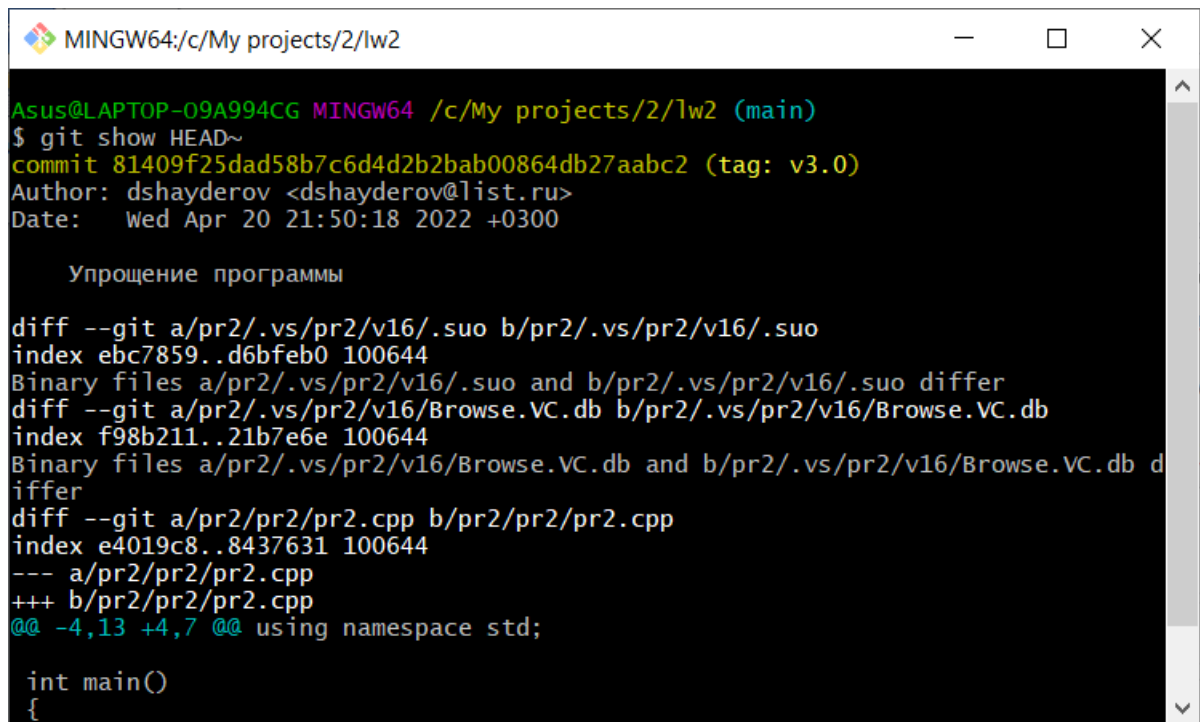
Рисунок 8 - История коммитов

7. Просмотрел содержимое коммитов командой `git show HEAD`, `git show HEAD~`, `git show 16b069d6578b6fcda59b508f4def0507fc237bf8`:

```
Asus@LAPTOP-09A994CG MINGW64 /c/My projects/2/lw2 (main)
$ git show HEAD
commit a16dc9aaad55127eb12405872b18a42801ef0713 (HEAD -> main, origin/main, orig
in/HEAD)
Merge: 81409f2 e966e8d
Author: dshayderov <dshayderov@list.ru>
Date: Wed Apr 20 21:52:12 2022 +0300

Merge branch 'main' of https://github.com/dshayderov/lw2
```

Рисунок 9 - git show HEAD



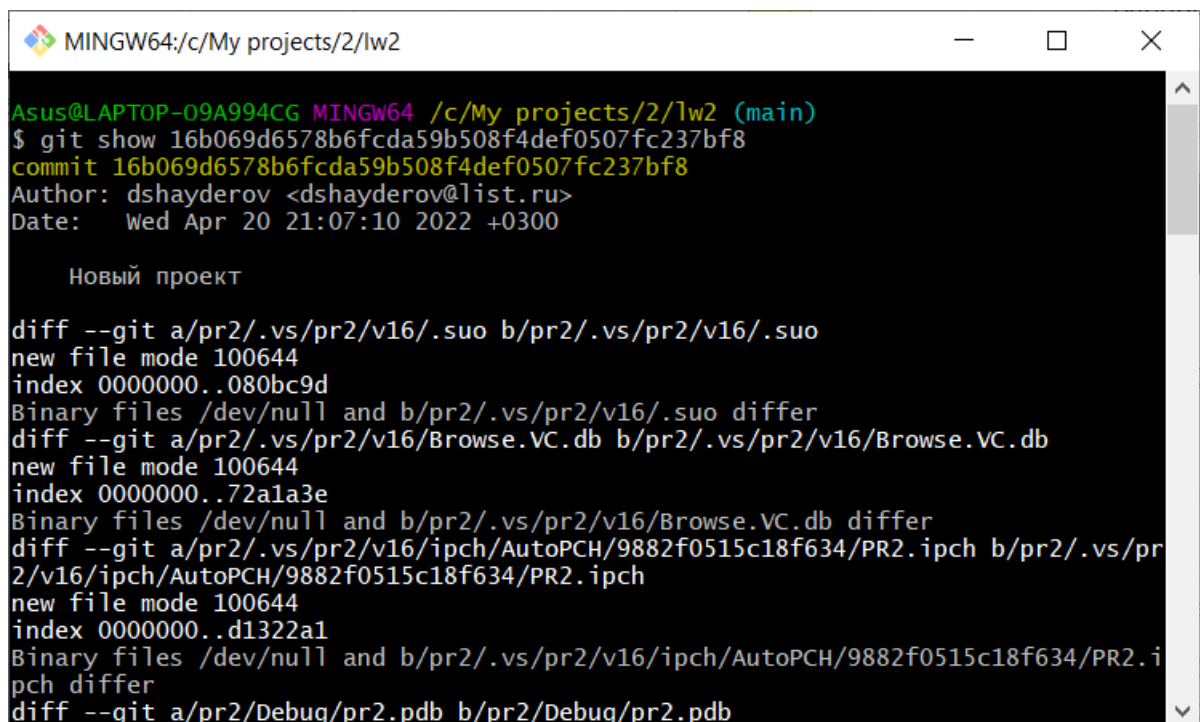
```
MINGW64:/c/My projects/2/lw2
Asus@LAPTOP-09A994CG MINGW64 /c/My projects/2/lw2 (main)
$ git show HEAD~
commit 81409f25dad58b7c6d4d2b2bab00864db27aabc2 (tag: v3.0)
Author: dshayderov <dshayderov@list.ru>
Date: Wed Apr 20 21:50:18 2022 +0300

    Упрощение программы

diff --git a/pr2/.vs/pr2/v16/.suo b/pr2/.vs/pr2/v16/.suo
index ebc7859..d6bfeb0 100644
Binary files a/pr2/.vs/pr2/v16/.suo and b/pr2/.vs/pr2/v16/.suo differ
diff --git a/pr2/.vs/pr2/v16/Browse.VC.db b/pr2/.vs/pr2/v16/Browse.VC.db
index f98b211..21b7e6e 100644
Binary files a/pr2/.vs/pr2/v16/Browse.VC.db and b/pr2/.vs/pr2/v16/Browse.VC.db differ
diff --git a/pr2/pr2/pr2.cpp b/pr2/pr2/pr2.cpp
index e4019c8..8437631 100644
--- a/pr2/pr2/pr2.cpp
+++ b/pr2/pr2/pr2.cpp
@@ -4,13 +4,7 @@ using namespace std;

int main()
{
```

Рисунок 10 - git show HEAD



```
MINGW64:/c/My projects/2/lw2
Asus@LAPTOP-09A994CG MINGW64 /c/My projects/2/lw2 (main)
$ git show 16b069d6578b6fcda59b508f4def0507fc237bf8
commit 16b069d6578b6fcda59b508f4def0507fc237bf8
Author: dshayderov <dshayderov@list.ru>
Date: Wed Apr 20 21:07:10 2022 +0300

    Новый проект

diff --git a/pr2/.vs/pr2/v16/.suo b/pr2/.vs/pr2/v16/.suo
new file mode 100644
index 0000000..080bc9d
Binary files /dev/null and b/pr2/.vs/pr2/v16/.suo differ
diff --git a/pr2/.vs/pr2/v16/Browse.VC.db b/pr2/.vs/pr2/v16/Browse.VC.db
new file mode 100644
index 0000000..72a1a3e
Binary files /dev/null and b/pr2/.vs/pr2/v16/Browse.VC.db differ
diff --git a/pr2/.vs/pr2/v16/ipch/AutoPCH/9882f0515c18f634/PR2.ipch b/pr2/.vs/pr2/v16/ipch/AutoPCH/9882f0515c18f634/PR2.ipch
new file mode 100644
index 0000000..d1322a1
Binary files /dev/null and b/pr2/.vs/pr2/v16/ipch/AutoPCH/9882f0515c18f634/PR2.ipch differ
diff --git a/pr2/Debug/pr2.pdb b/pr2/Debug/pr2.pdb
```

Рисунок 11 - git show 16b069d6578b6fcda59b508f4def0507fc237bf8

8. Удалил весь код в файле Untitled-1.cpp и сохранил его, затем удалил все несохраненные изменения командой, после этого еще раз удалил весь код в файле и сделал коммит, после чего откатил состояние файла к предыдущей версии.

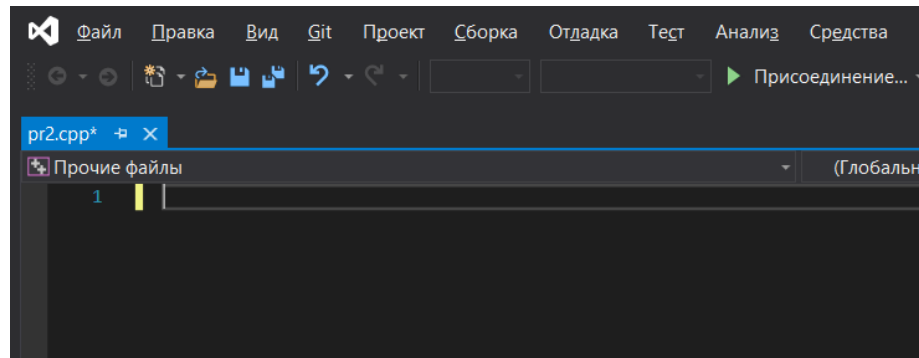


Рисунок 12 - Удаление кода из файла pr2.cpp

```

MINGW64:/c/My projects/2/lw2/pr2/pr2
Asus@LAPTOP-09A994CG MINGW64 /c/My projects/2/lw2/pr2/pr2 (main)
$ git checkout -- pr2.cpp

Asus@LAPTOP-09A994CG MINGW64 /c/My projects/2/lw2/pr2/pr2 (main)
$ |

```

Рисунок 13 - Удаление несохраненных изменений при помощи команды git checkout

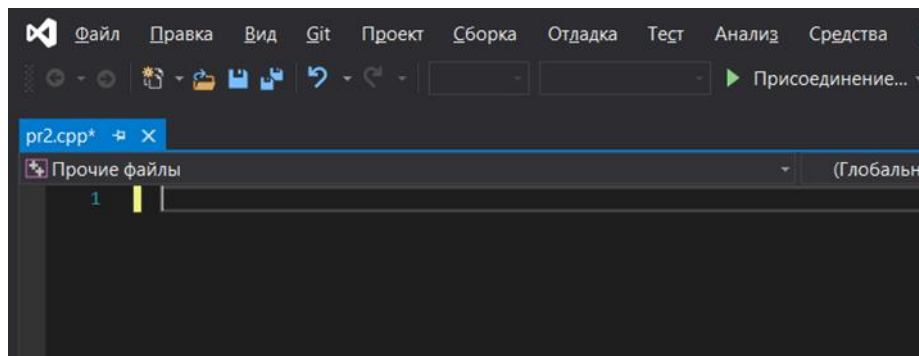


Рисунок 14 - Удаление кода

```

MINGW64:/c/My projects/2/lw2
Asus@LAPTOP-09A994CG MINGW64 /c/My projects/2/lw2 (main)
$ git add .

Asus@LAPTOP-09A994CG MINGW64 /c/My projects/2/lw2 (main)
$ git commit --m "Пункт 10"
[main 7a1ddcf] пункт 10
3 files changed, 1 insertion(+), 36 deletions(-)
rewrite pr2/.vs/pr2/v16/.suo (62%)
rewrite pr2/pr2/pr2.cpp (100%)

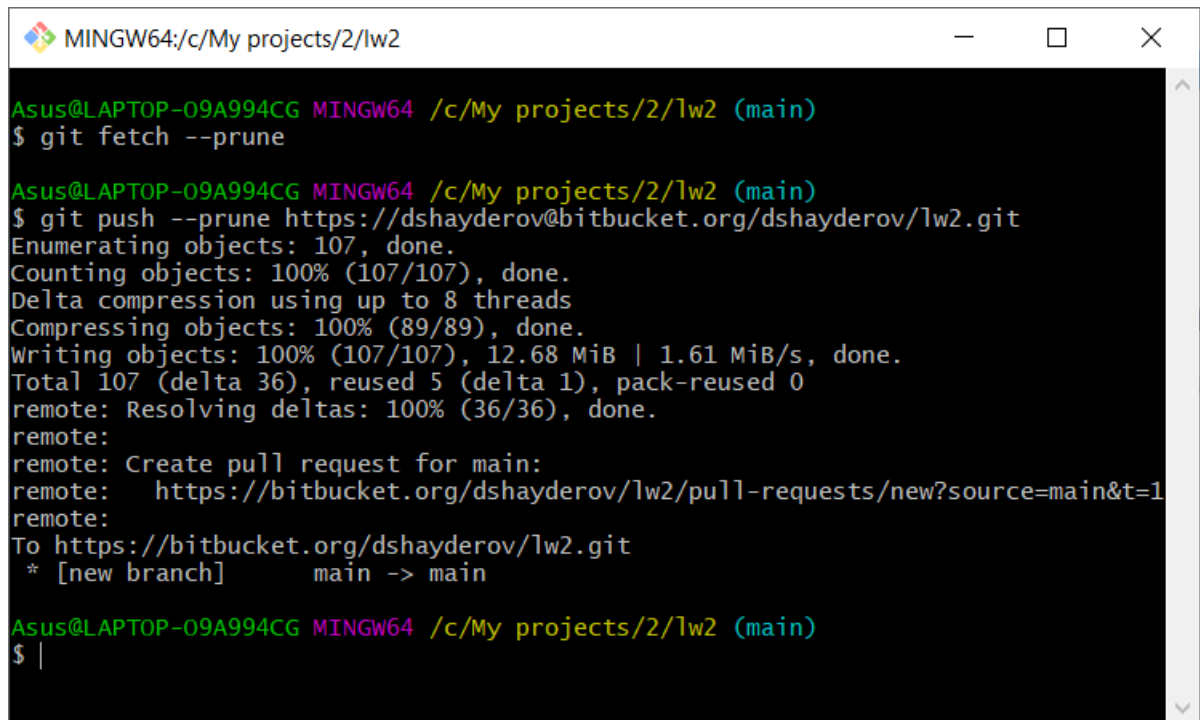
Asus@LAPTOP-09A994CG MINGW64 /c/My projects/2/lw2 (main)
$ git reset --hard HEAD~1
HEAD is now at a16dc9a Merge branch 'main' of https://github.com/dshayderov/lw2

```

Рисунок 15 - Коммит и команда git reset --hard HEAD~1

команда git --checkout удаляет изменения произошедшие с файлом в репозитории до коммита.

9. Создал зеркало репозитория на BitBucket.



```
MINGW64:/c/My projects/2/lw2
Asus@LAPTOP-09A994CG MINGW64 /c/My projects/2/lw2 (main)
$ git fetch --prune

Asus@LAPTOP-09A994CG MINGW64 /c/My projects/2/lw2 (main)
$ git push --prune https://dshayderov@bitbucket.org/dshayderov/lw2.git
Enumerating objects: 107, done.
Counting objects: 100% (107/107), done.
Delta compression using up to 8 threads
Compressing objects: 100% (89/89), done.
Writing objects: 100% (107/107), 12.68 MiB | 1.61 MiB/s, done.
Total 107 (delta 36), reused 5 (delta 1), pack-reused 0
remote: Resolving deltas: 100% (36/36), done.
remote:
remote: Create pull request for main:
remote:   https://bitbucket.org/dshayderov/lw2/pull-requests/new?source=main&t=1
remote:
To https://bitbucket.org/dshayderov/lw2.git
 * [new branch]      main -> main

Asus@LAPTOP-09A994CG MINGW64 /c/My projects/2/lw2 (main)
$ |
```

Рисунок 16 - Команды git fetch --prune и git push --prune

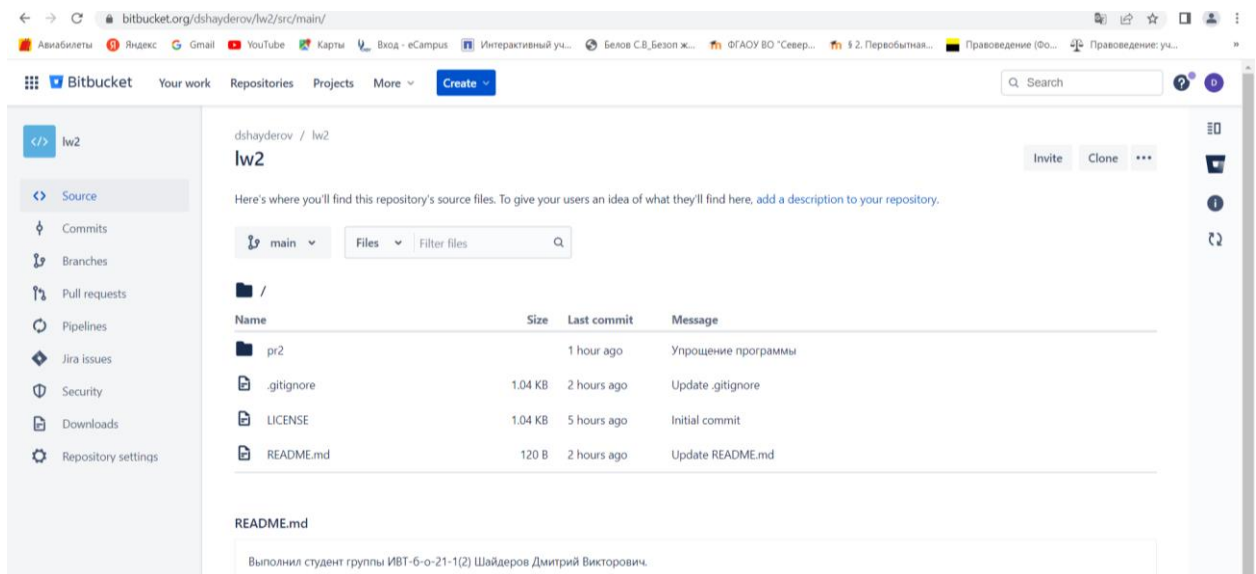


Рисунок 17 - Репозиторий на BitBucket

Адрес репозитория: <https://bitbucket.org/dshayderov/lw2/src/main/>

Ответы на контрольные вопросы:

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 данной 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`. Эта 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`.

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` напомнит вам: Прямо под текстом «Changes to be committed» говорится: используйте `git reset HEAD` для исключения из индекса.

5. Как отменить изменения в файле?

С помощью команды `git checkout --`.

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 репозиториях, 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`.

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`. 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`.

10. Как выполнить просмотр удаленного репозитория?

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

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`. Для отправки всех тегов можно использовать команду `git push origin tags`.

Для удаления тега в локальной репозитории достаточно выполнить команду `git tag -d`. Например, удалить созданный ранее легко весный тег можно следующим образом: `git tag -d v1.4-lw`. Для удаления тега из внешнего репозитория используется команда `git push origin --delete`.

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.

`Git push --prune` позволяет выполнить отправку содержимого локального репозитория в репозиторий GitLab или BitBucket.

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