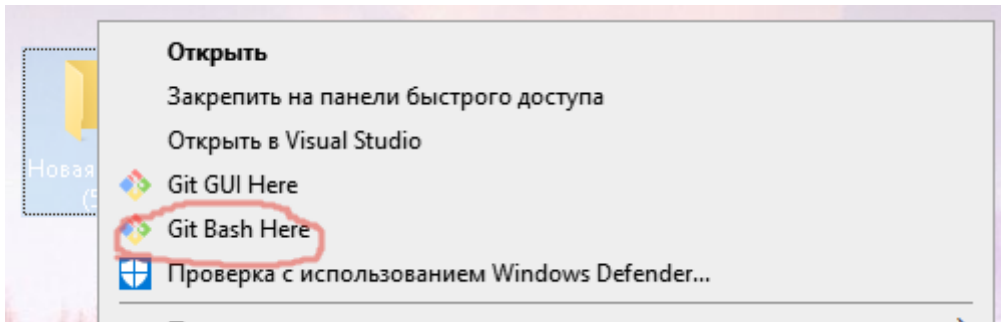


Лабораторная №1. Работа с Git

1. Создать репозиторий.

Чтобы создать репозиторий в git нам нужно создать папку и после нажать на неё правой кнопкой мыши и выбрать “Git Bash Here”.



После появится командная строка GIT, в ней нужно прописать команду “git init”, после этого на нашем компьютере создастся репозиторий

```
MINGW64/c:/Users/Пекарня Александра/Desktop/Новая папка (5)
Пекарня Александра@DESKTOP-1K95JQE MINGW64 ~/Desktop/Новая папка (5) (master)
$ git init
Initialized empty Git repository in C:/Users/Пекарня Александра/Desktop/Новая папка (5)/.git/
Пекарня Александра@DESKTOP-1K95JQE MINGW64 ~/Desktop/Новая папка (5) (master)
$ |
```

2. Добавить файлы к отслеживанию.

Чтобы добавить файлы к отслеживанию, нам нужно в командной строке Gita прописать команду “git add <имя файла>”

```
Пекарня Александра@DESKTOP-1K95JQE MINGW64 ~/Desktop/Новая папка (5) (master)
$ git add Example.txt
Пекарня Александра@DESKTOP-1K95JQE MINGW64 ~/Desktop/Новая папка (5) (master)
$ |
```

3. Просмотреть состояние репозитория.

Чтобы проверить состояние репозитория нужно прописать команду “**git status**”

```
Пекарня Александра@DESKTOP-1K9SJJQE MINGW64 ~/Desktop/Новая папка (5) (master)
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   Example.txt

Пекарня Александра@DESKTOP-1K9SJJQE MINGW64 ~/Desktop/Новая папка (5) (master)
$
```

4. Сделать коммит (зафиксировать изменения).

Чтобы сделать коммит или зафиксировать изменения нам нужно прописать команду “**git commit -am “<Сообщение коммита>”**”

```
Пекарня Александра@DESKTOP-1K9SJJQE MINGW64 ~/Desktop/Новая папка (5) (master)
$ git commit -am "example"
[master (root-commit) a285dcd] example
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 Example.txt

Пекарня Александра@DESKTOP-1K9SJJQE MINGW64 ~/Desktop/Новая папка (5) (master)
$ |
```

5. Исправить сообщение предыдущего коммита.

Чтобы исправить сообщение предыдущего коммита нам нужно прописать команду “**git commit --amend -m “<Новое сообщение коммита>”**”

```
Пекарня Александра@DESKTOP-1K9SJJQE MINGW64 ~/Desktop/Новая папка (5) (master)
$ git commit --amend -m "New example"
[master e7b6328] New example
Date: Sun Oct 18 16:35:04 2020 +0900
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 Example.txt

Пекарня Александра@DESKTOP-1K9SJJQE MINGW64 ~/Desktop/Новая папка (5) (master)
$ |
```

6. Сделать коммит, посмотреть разницу между коммитами.

Чтобы посмотреть изменения между коммитами нам нужно прописать команду “**git log -p**”. Перед этим конечно же создадим новый файл и сделаем новый коммит.

```
MINGW64:/c:/Users/Пекарня Александра/Desktop/Новая папка (5)
Пекарня Александра@DESKTOP-1K9SJJQE MINGW64 ~/Desktop/Новая папка (5) (master)
$ git log -p
commit c632ab9e615ae60570726efab0bf8ba4825ab759 (HEAD -> master)
Author: NineTrier <rarnenskiy.2015@mail.ru>
Date: Sun Oct 18 16:47:21 2020 +0900

    Second example

diff --git a/Example2.txt b/Example2.txt
new file mode 100644
index 0000000..e69de29

commit e7b632856e2d9a2134e2e98f6f54f1f5ca6397eb
Author: NineTrier <rarnenskiy.2015@mail.ru>
Date: Sun Oct 18 16:35:04 2020 +0900

    New example

diff --git a/Example.txt b/Example.txt
new file mode 100644
index 0000000..e69de29

Пекарня Александра@DESKTOP-1K9SJJQE MINGW64 ~/Desktop/Новая папка (5) (master)
$ |
```

7. Создать новую ветку.

Чтобы создать новую ветку нам нужно прописать команду **“git branch <имя ветки>”**

```
Пекарня Александра@DESKTOP-1K9SJJQE MINGW64 ~/Desktop/Новая папка (5) (master)
$ git branch example

Пекарня Александра@DESKTOP-1K9SJJQE MINGW64 ~/Desktop/Новая папка (5) (master)
$
```

8. Переключиться на новую ветку.

Чтобы переключиться на новую ветку нам нужно прописать команду **“git checkout <Имя ветки>”**

```
Пекарня Александра@DESKTOP-1K9SJJQE MINGW64 ~/Desktop/Новая папка (5) (master)
$ git checkout example
Switched to branch 'example'

Пекарня Александра@DESKTOP-1K9SJJQE MINGW64 ~/Desktop/Новая папка (5) (example)
$
```

9. Внести изменения. Посмотреть разницу (diff) между ветками.

В новой ветке откроем файл, который уже присутствовал в коммите в основной ветке и сделаем коммит. После переключимся на основную ветку при помощи команды `git checkout master`.

```

Пекарня Александра@DESKTOP-1K9SJQE MINGW64 ~/Desktop/Новая папка (5) (example)
$ git add Example.txt

Пекарня Александра@DESKTOP-1K9SJQE MINGW64 ~/Desktop/Новая папка (5) (example)
$ git commit -am "izmenenia"
[example f1b6f31] izmenenia
1 file changed, 1 insertion(+)

Пекарня Александра@DESKTOP-1K9SJQE MINGW64 ~/Desktop/Новая папка (5) (example)
$ git checkout

Пекарня Александра@DESKTOP-1K9SJQE MINGW64 ~/Desktop/Новая папка (5) (example)
$ git checkout master
Switched to branch 'master'

Пекарня Александра@DESKTOP-1K9SJQE MINGW64 ~/Desktop/Новая папка (5) (master)

```

Теперь пропишем команду для отслеживания изменений между ветками “**git diff <имя ветки с которой сравниваем>**”

```

Пекарня Александра@DESKTOP-1K9SJQE MINGW64 ~/Desktop/Новая папка (5) (master)
$ git diff example
diff --git a/Example.txt b/Example.txt
index 8c9ed77..e69de29 100644
--- a/Example.txt
+++ b/Example.txt
@@ -1,0 @@
-Changes was maked
\ No newline at end of file
diff --git a/ExampleBranch.txt b/ExampleBranch.txt
deleted file mode 100644
index e69de29..0000000

Пекарня Александра@DESKTOP-1K9SJQE MINGW64 ~/Desktop/Новая папка (5) (master)

```

10. Зафиксировать изменения.

Зафиксируем изменения в главной ветке командой “**git commit -am <Сообщение к коммиту>**”.

```

@@ -1,0 @@
-Changes was maked
\ No newline at end of file
diff --git a/ExampleBranch.txt b/ExampleBranch.txt
deleted file mode 100644
index e69de29..0000000

Пекарня Александра@DESKTOP-1K9SJQE MINGW64 ~/Desktop/Новая папка (5) (master)
$ git commit -am "Changes"
On branch master
nothing to commit, working tree clean

Пекарня Александра@DESKTOP-1K9SJQE MINGW64 ~/Desktop/Новая папка (5) (master)
$ git add Example.txt

Пекарня Александра@DESKTOP-1K9SJQE MINGW64 ~/Desktop/Новая папка (5) (master)
$ git commit -am "Changes"
On branch master
nothing to commit, working tree clean

```

11. Посмотреть что находится в файлах на ветке master.

Для этого на ветке мастер в файлах напишем какой-нибудь текст. После создадим новую ветку, и при помощи команды “**git diff**” проверим что содержится в файлах на ветке master.

```

Пекарня Александра@DESKTOP-1K9SJQE MINGW64 ~/Desktop/Новая папка (5) (master)
$ git branch example2

Пекарня Александра@DESKTOP-1K9SJQE MINGW64 ~/Desktop/Новая папка (5) (master)
$ git checkout example2
Switched to branch 'example2'
M       Example.txt
M       Example2.txt

Пекарня Александра@DESKTOP-1K9SJQE MINGW64 ~/Desktop/Новая папка (5) (example2)
$ git diff master
diff --git a/Example.txt b/Example.txt
index e69de29..e965047 100644
--- a/Example.txt
+++ b/Example.txt
@@ -0,0 +1 @@
+Hello
diff --git a/Example2.txt b/Example2.txt
index e69de29..1841c97 100644
--- a/Example2.txt
+++ b/Example2.txt
@@ -0,0 +1 @@
+Okey
\ No newline at end of file

Пекарня Александра@DESKTOP-1K9SJQE MINGW64 ~/Desktop/Новая папка (5) (example2)
$

```

12. Объединить ветки.

В данный момент мы находимся на ветке example2, напомним команду “**git merge <имя ветки с которой хотим объединиться>**”, и объединим её с веткой master.

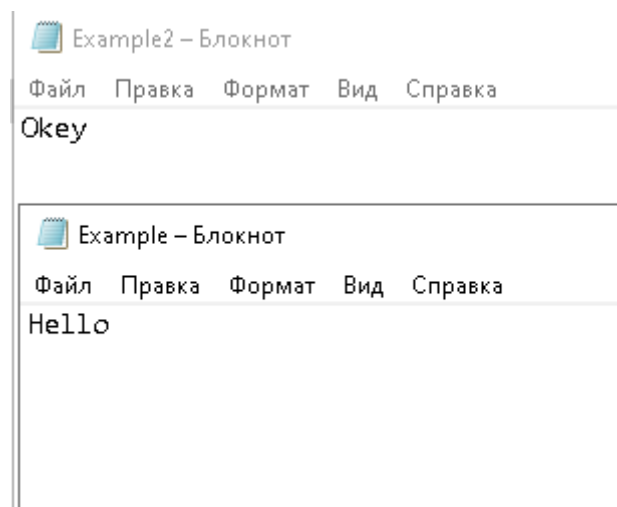
```

Пекарня Александра@DESKTOP-1K9SJQE MINGW64 ~/Desktop/Новая папка (5) (example2)
$ git merge master
Already up to date.

Пекарня Александра@DESKTOP-1K9SJQE MINGW64 ~/Desktop/Новая папка (5) (example2)
$ |

```

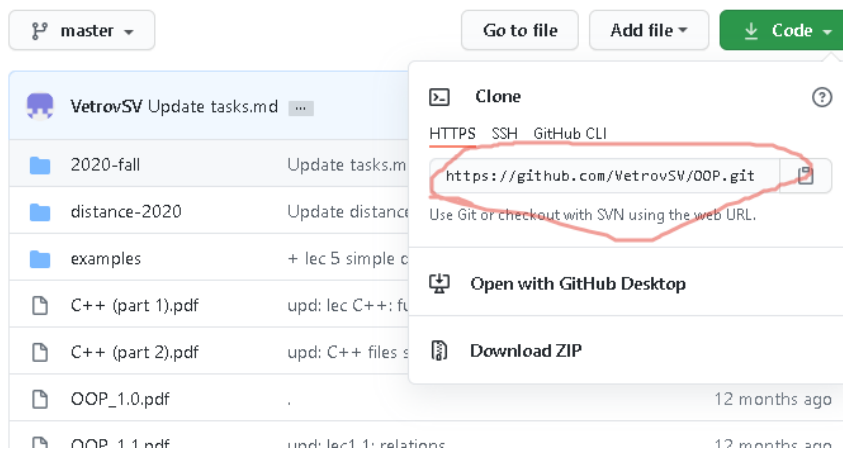
Данные с ветки master перешли на ветку example2



13. Клонировать удалённый репозиторий.

Чтобы клонировать репозиторий, нужно на платформе GitHub выбрать репозиторий и

нажать на кнопку Code и в списке найти ссылку и скопировать её. Я буду использовать репозиторий с нашего занятия



После в gite нужно будет прописать команду “**git clone <Ссылка на удалённый репозиторий>**”

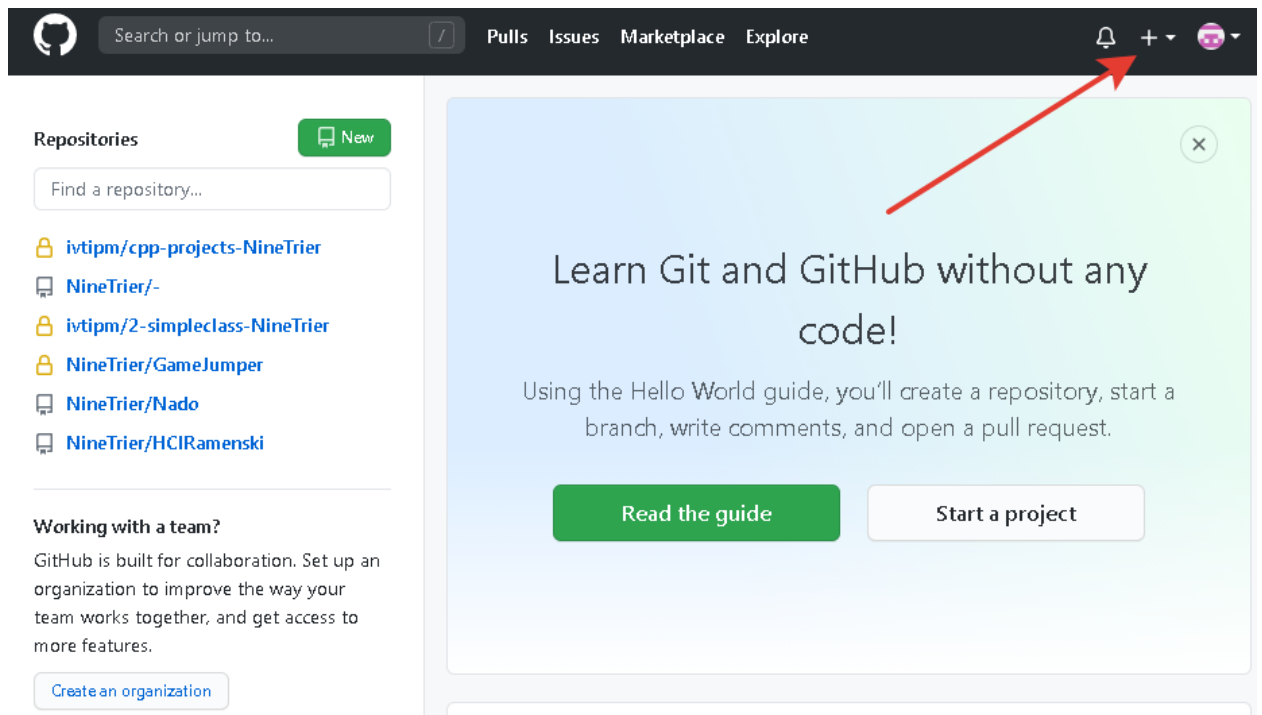
```
Пекарня Александра@DESKTOP-1K9S3JQE MINGW64 ~/Desktop/Новая папка (5) (master)
$ git clone https://github.com/VetrovSV/OOP.git
Cloning into 'OOP'...
remote: Enumerating objects: 162, done.
remote: Counting objects: 100% (162/162), done.
remote: Compressing objects: 100% (125/125), done.
Receiving objects: 23% (228/988), 10.50 MiB | 116.00 KiB/s
```

После начнётся клонирование репозитория, которое может быть долгим в зависимости от кол-ва файлов на репозитории.

14. Отправить изменения в удалённый репозиторий, забрать изменения из удалённого репозитория.

Чтобы отправить изменения на удалённый репозиторий, нужно сначала создать его.

Нажмите на плюсик в верхнем правом углу сайта и придумайте название для репозитория.



После копируем ссылку на репозиторий. Далее пишем в консоли Git команду

“git remote add origin <ссылка на репозиторий>”

```
Пекарня Александра@DESKTOP-1K9SJQE MINGW64 ~/Desktop/Новая папка (5) (master)
$ git remote add origin https://github.com/NineTrier/Example.git
```

После проводим коммит и пишем ещё одну новую команду **“git push <ссылка на репозиторий>”**

```
Пекарня Александра@DESKTOP-1K9SJQE MINGW64 ~/Desktop/Новая папка (5) (master)
$ git commit -am "Oops"
[master 6ccc587] Oops
 3 files changed, 2 insertions(+)
 create mode 100644 ForRepository.txt

Пекарня Александра@DESKTOP-1K9SJQE MINGW64 ~/Desktop/Новая папка (5) (master)
Enumerating objects: 9, done./NineTrier/Example.git
Counting objects: 100% (9/9), done.
Delta compression using up to 8 threads
Compressing objects: 100% (5/5), done.
Writing objects: 100% (9/9), 746 bytes | 373.00 KiB/s, done.
Total 9 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/NineTrier/Example.git
 * [new branch]      master -> master

Пекарня Александра@DESKTOP-1K9SJQE MINGW64 ~/Desktop/Новая папка (5) (master)
$ git push https://github.com/NineTrier/Example.git
```

Готово на удалённом репозитории файлы, которые мы отслеживали

NineTrier / Example

Unwatch
1

Code
Issues
Pull requests
Actions
Projects
Wiki
Security

master
Go to file
Add file
Code

NineTrier
Oops
6 minutes ago
3

Example.txt	Oops	6 minutes ago
Example2.txt	Oops	6 minutes ago
ForRepository.txt	Oops	6 minutes ago

Help people interested in this repository understand your project by adding a README.
Add a README

А чтобы забрать изменения из удалённого репозитория, нужно прописать команду “**git pull <ссылка на удалённый репозиторий> <название ветки, на которую хотим загрузить изменения>**”.

Изменим 1 из файлов

NineTrier / Example

Unwatch
1
Star
0
Fork
0

Code
Issues
Pull requests
Actions
Projects
Wiki
Security
Insights

Example / Example.txt
Cancel

Edit file
Preview changes

Spaces
2
Soft wrap

1 Hello112414

На

NineTrier / Example

Unwatch 1 Star 0 Fork 0

<> Code

Issues

Pull requests

Actions

Projects

Wiki

Security

Insights

...

master Example / Example.txt

Go to file ...

NineTrier Haha

Latest commit ef6796c 1 minute ago History

1 contributor

1 lines (1 sloc) 15 Bytes

Raw Blame

📄 ✎ 🗑

1 It's not hello

Теперь прописываем команду в консоли

```
Пекарня Александра@DESKTOP-1K9SJQE MINGW64 ~/Desktop/Новая папка (5) (master)
$ git pull https://github.com/NineTrier/Example.git master
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), 695 bytes | 3.00 KiB/s, done.
From https://github.com/NineTrier/Example
 * branch          master      -> FETCH_HEAD
Updating 6ccc587..ef6796c
Fast-forward
 Example.txt | 2 +-
 1 file changed, 1 insertion(+), 1 deletion(-)

Пекарня Александра@DESKTOP-1K9SJQE MINGW64 ~/Desktop/Новая папка (5) (master)
$
```

Проверяем

Example – Блокнот

Файл Правка Формат Вид Справка

It's not hello

Изменения загрузились.

15. Создать worktree для ветки master.

Для создания нового worktree нужно прописать команду “**git worktree add -b <название новой ветки> <название рабочего дерева><путь к новому рабочему дереву>**”

```
Пекарня Александра@DESKTOP-1K9S3QE MINGW64 ~/Desktop/Новая папка (5) (master)
$ git worktree add -b v0 yes
Preparing worktree (new branch 'v0')
HEAD is now at ef6796c Haha
```

После в локальном репозитории появится папка с именем, которое мы указали в **<путь к новому рабочему дереву>**, а в самом Gite рабочее дерево будет называться так как мы его назвали в **<название рабочего дерева>**

```
Пекарня Александра@DESKTOP-1K9S3QE MINGW64 ~/Desktop/Новая папка (5) (master)
$ git log
commit ef6796c30c8a2fb188a13fdb087aa47ba496b303 (HEAD -> master, v0)
Author: NineTrier <63592407+NineTrier@users.noreply.github.com>
Date:   Sun Oct 18 19:30:27 2020 +0900

    Haha
```

Имя	Дата изменения	Тип	Размер
yes	18.10.2020 20:00	Папка с файлами	
Example	18.10.2020 19:31	Текстовый докум...	1 КБ
Example2	18.10.2020 17:56	Текстовый докум...	1 КБ
ForRepository	18.10.2020 18:12	Текстовый докум...	0 КБ

Моделирование конфликта. Решение данного конфликта.

Смоделируем конфликт двух веток, когда в файлах зафиксированы изменения в одном и том же месте. Для этого, создадим новый файл

```
Пекарня Александра@DESKTOP-1K9SJQE MINGW64 ~/Desktop/Новая папка (5) (master)
$ git add TestConflict.txt

Пекарня Александра@DESKTOP-1K9SJQE MINGW64 ~/Desktop/Новая папка (5) (master)
$ git commit -am "MergeTest"
[master 825be4a] MergeTest
 1 file changed, 1 insertion(+)
 create mode 100644 TestConflict.txt
```

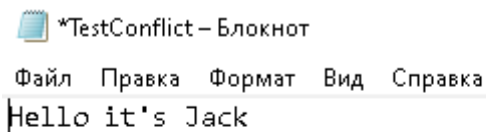
Теперь создадим новую ветку и перейдём к ней

```
Пекарня Александра@DESKTOP-1K9SJQE MINGW64 ~/Desktop/Новая папка (5) (master)
$ git branch TestMerge

Пекарня Александра@DESKTOP-1K9SJQE MINGW64 ~/Desktop/Новая папка (5) (master)
$ git checkout TestMerge
Switched to branch 'TestMerge'

Пекарня Александра@DESKTOP-1K9SJQE MINGW64 ~/Desktop/Новая папка (5) (TestMerge)
$
```

Снова открываем файл и изменяем его, а также проводим коммит



*TestConflict – Блокнот

Файл Правка Формат Вид Справка

Hello it's Jack

```
Пекарня Александра@DESKTOP-1K9SJQE MINGW64 ~/Desktop/Новая папка (5) (TestMerge)
$ git add TestConflict.txt

Пекарня Александра@DESKTOP-1K9SJQE MINGW64 ~/Desktop/Новая папка (5) (TestMerge)
$ git commit -am "TestMerge"
[TestMerge 5193723] TestMerge
 1 file changed, 1 insertion(+), 1 deletion(-)
```

После переходим обратно в главную ветку меняем содержимое файла, проводим коммит и объединяем ветки

TestConflict – Блокнот

Файл Правка Формат Вид Справка

Hello it's Pavel

```
Пекарня Александра@DESKTOP-1K9S3QE MINGW64 ~/Desktop/Новая папка (5) (TestMerge)
$ git checkout master
Switched to branch 'master'

Пекарня Александра@DESKTOP-1K9S3QE MINGW64 ~/Desktop/Новая папка (5) (master)
$ git add TestConflict.txt

Пекарня Александра@DESKTOP-1K9S3QE MINGW64 ~/Desktop/Новая папка (5) (master)
$ git commit -am "TestMerge"
[master 223c026] TestMerge
1 file changed, 1 insertion(+), 1 deletion(-)

Пекарня Александра@DESKTOP-1K9S3QE MINGW64 ~/Desktop/Новая папка (5) (master)
$ git merge TestMerge
Auto-merging TestConflict.txt
CONFLICT (content): Merge conflict in TestConflict.txt
Automatic merge failed; fix conflicts and then commit the result.

Пекарня Александра@DESKTOP-1K9S3QE MINGW64 ~/Desktop/Новая папка (5) (master|MER
GING)
```

TestConflict – Блокнот

Файл Правка Формат Вид Справка

<<<<<<< HEAD
Hello it's Jack
=====
Hello it's Pavel
>>>>>> TestMerge

Появился конфликт. Теперь чтобы его решить мы можем в ручную залезть в файл и переделать файл, после провести коммит.

TestConflict – Блокнот

Файл Правка Формат Вид Справка

Hello it's Jack Pavel

```

Пекарня Александра@DESKTOP-1K9SJQE MINGW64 ~/Desktop/Новая папка (5) (master|MER
GING)
$ git add TestConflict.txt

Пекарня Александра@DESKTOP-1K9SJQE MINGW64 ~/Desktop/Новая папка (5) (master|MER
GING)
$ git commit -am "TestMerge"
[master 3750cc8] TestMerge

Пекарня Александра@DESKTOP-1K9SJQE MINGW64 ~/Desktop/Новая папка (5) (master)
$ |

```

Использование тегов.

Git имеет возможность пометить определённые моменты в истории как важные. Чтобы посмотреть список имеющихся тегов нужно прописать команду **“git tag”** (-l,--list).

Создание меток.

В Git есть 2 типа меток: легковесные и аннотированные.

Легковесный тег — это что-то очень похожее на ветку, которая не изменяется — просто указатель на определённый коммит.

А вот аннотированные теги хранятся в базе данных Git как полноценные объекты. Они имеют контрольную сумму, содержат имя автора, его e-mail и дату создания, имеют комментарий и могут быть подписаны и проверены с помощью GNU Privacy Guard (GPG).

Аннотированные метки

Чтобы создать аннотированную метку нужно прописать команду **“git tag -a “<Название тега>” -m “<сообщение к тегу>”**

```

Пекарня Александра@DESKTOP-1K9SJQE MINGW64 ~/Desktop/Новая папка (5) (master)
$ git tag -a "v1.4" -m "Version 1.4"

Пекарня Александра@DESKTOP-1K9SJQE MINGW64 ~/Desktop/Новая папка (5) (master)
$ git tag
v1.4

```

Легковесные метки

Чтобы создать легковесную метку нужно прописать команду **“git tag <название тега>”**

```

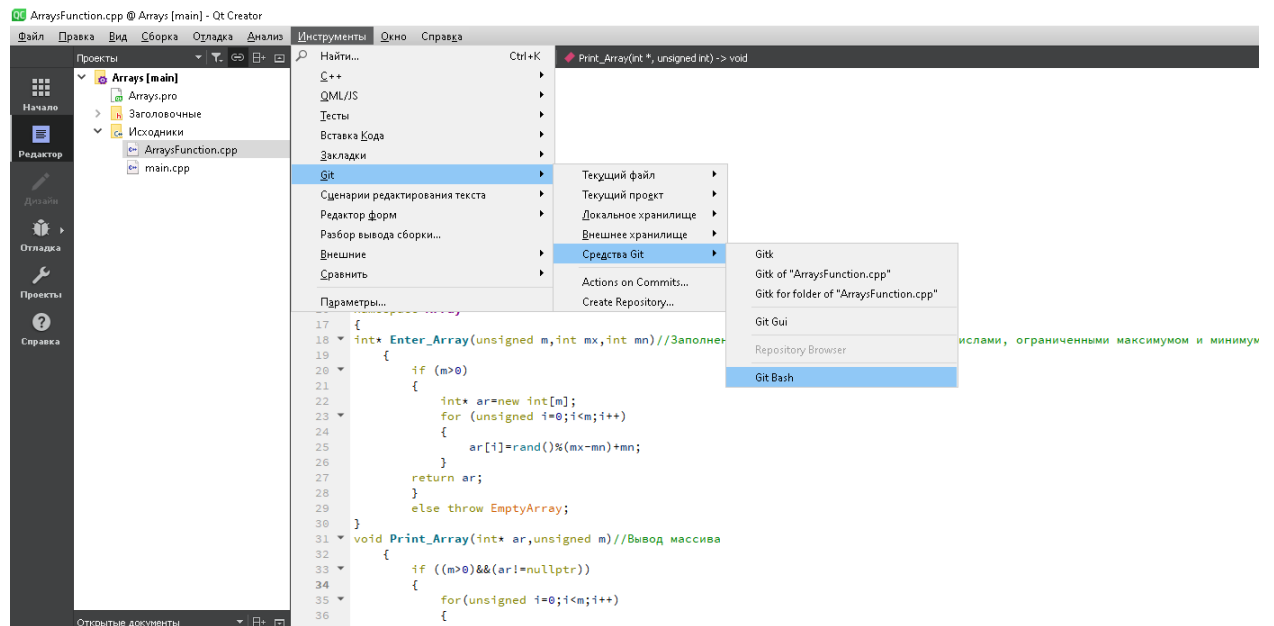
Пекарня Александра@DESKTOP-1K9SJQE MINGW64 ~/Desktop/Новая папка (5) (master)
$ git tag v1.5

Пекарня Александра@DESKTOP-1K9SJQE MINGW64 ~/Desktop/Новая папка (5) (master)
$ git tag
v1.4
v1.5

```

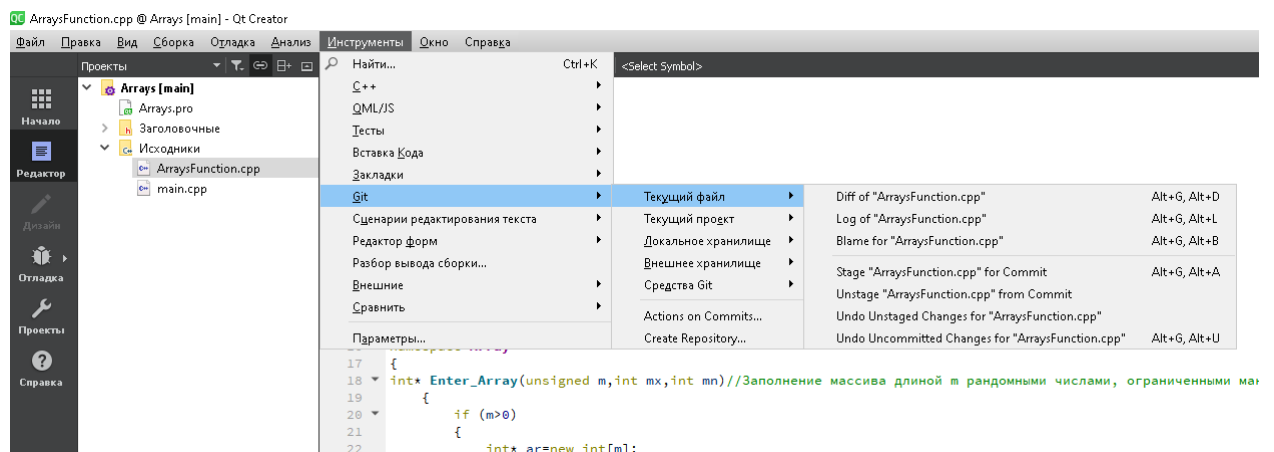
16. Git в Qt Creator

В Qt Creator можно работать со встроенным Git, для этого нужно перейти **Инструменты->Git->Средства Git->Git Bush**. Это откроет консоль Git прямо в Qt Creator.



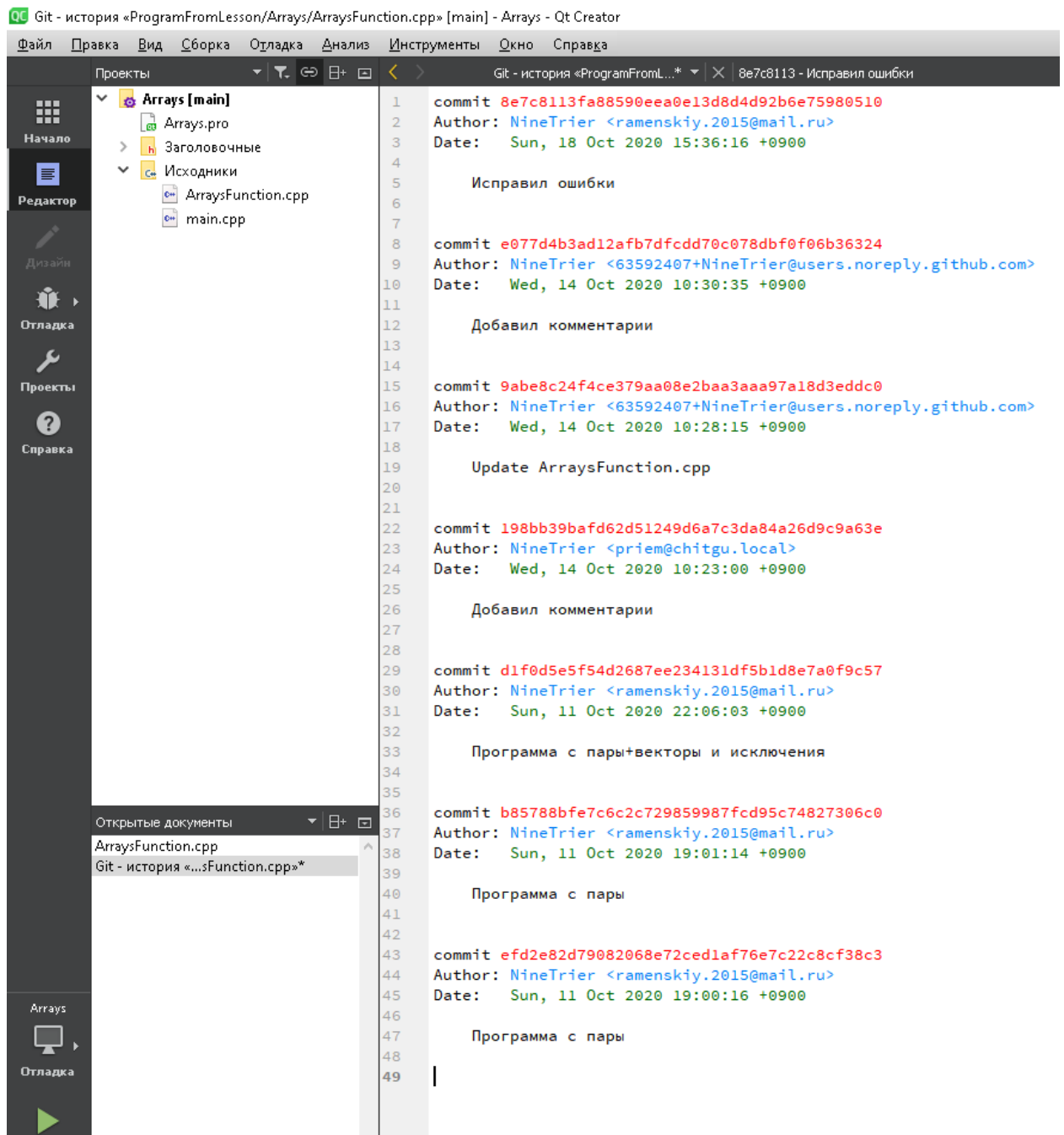
Дальше вся работа похожа на работу с обычным Git.

Также в Qt Creator можно посмотреть состояние, лог и другие свойства текущего файла или проекта, для этого надо также перейти в подменю Git и оттуда в текущий файл или проект.



После можно посмотреть diff, log и другое данного файла или проекта.

Пример лога в Qt Creator



Вопросы:

1. Что такое система управления версиями? Для чего она используется?

Ответ: Система управления версиями – программа, которая запоминает все модификации данных, и при необходимости позволяет выполнить откат. Используется для облегчения хранения версий вашего проекта, а также для удобства передачи этих версий между участниками данного проекта.

2. Что такое репозиторий (локальный и удалённый)?

Ответ: Репозиторий это место, в котором хранятся данные вашего проекта, за которыми следит система управления версиями. Бывает 2-ух видов: локальный и удалённый.

Локальный репозиторий это репозиторий, который находится на вашем компьютере, к примеру обычная папка и есть репозиторий. Удалённый репозиторий это место для хранения ваших данных в облаке, например на GitHub.

3. Какие файлы следует добавлять к отслеживанию, а какие нет.

Ответ:

К отслеживанию следует добавлять текстовые файлы, которые мы собираемся изменять, если мы не собираемся их менять, то и добавлять не следует. Не следует же добавлять разного рода видео, изображения, приложения и т.д.

4. В каких случаях создавать ветку?

Ответ: Ветку следует создавать в тех случаях, когда вы хотите изменить файлы, но не хотите, чтобы они сразу же сохранились в главной ветке. Например, хотите разработать новую версию, создайте новую ветку с названием данной версии и спокойно работайте в ней с вашими файлами, после, если всё пройдёт успешно, можно объединить ветки.

Либо ветки можно создавать для каждой новой версии, чтобы разные ветки содержали разные версии.

5. Что такое конфликт? Как исправить?

Ответ: Конфликт слияния это ситуация, когда в двух объединяемых ветках (master и любая другая) в одном и том же файле в одном и том же месте разные данные. В этот момент возникает конфликт, который необходимо разрешить. Чтобы разрешить его, необходимо исправить возникший конфликт в файле и сделать коммит.