Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт перспективной инженерии
Департамент цифровых, робототехнических систем и электроники

# ОТЧЕТ
# ПО ЛАБОРАТОРНОЙ РАБОТЕ №2
## дисциплины
## «Основы кроссплатформенного программирования

Выполнил:
Мидов Андемир Исланович
2 курс, группа ИТС-б-о-23-1,
11.03.02 «Инфокоммуникационные технологии и системы связи»,
направленность (профиль) «Инфокоммуникационные системы и сети», очная форма обучения

_____
(подпись)

Проверил:

Воронкин Р. А.
_____
(подпись)

Отчет защищен с оценкой _____ Дата защиты_____

Ставрополь, 2024 г.

**Тема**: Исследование основных возможностей Git и GitHub

**Цель:** исследовать базовые возможности системы контроля версий Git и веб-сервиса для хостинга IT-проектов GitHub.

## Порядок выполнения работы:

1. Создал новый репозитории.

```
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   README.md


C:\Users\AND\laba2>git commit -m "Добавил информацию в README"
[main 49d318f] Добавил информацию в README
 1 file changed, 3 insertions(+), 1 deletion(-)

C:\Users\AND\laba2>git push
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 8 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 389 bytes | 389.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/midand-hash/laba2.git
   6092dea..49d318f  main -> main

C:\Users\AND\laba2>
```

Рисунок 1. Новый репозитории

2. Использование команды git log на репозитории

```
C:\Users\AND\laba2>git log
commit 66f72012647f658ae9a22186e2c970df36bbe9b2 (HEAD -> main, origin/main, origin/HEAD)
Author: midand-hash <midovandik07@gmail.com>
Date:   Thu Dec 26 17:03:53 2024 +0300

    Внёс изменения для точности кода

commit 96c0441e2ef410bada2cef6781b24af72d3700b7
Author: midand-hash <midovandik07@gmail.com>
Date:   Thu Dec 26 17:00:03 2024 +0300

    Написал код в новом файле

commit d7fbf637059846d8e713918404b6c126e91172fa
Author: midand-hash <midovandik07@gmail.com>
Date:   Thu Dec 26 16:55:24 2024 +0300

    Добавил новый пустой файл

commit 49d318f94736b94c7c4d30e997c357b4632e2bf6
Author: midand-hash <midovandik07@gmail.com>
Date:   Thu Dec 26 16:52:31 2024 +0300

    Добавил информацию в README

commit 6092dead549482ba30be360549bfc37459be9b7f
Author: midand-hash <midovandik07@gmail.com>
Date:   Thu Dec 26 16:42:12 2024 +0300

    Initial commit
:...skipping...
```

Рисунок.2 Результат выполнения команды git log

3. Использование команды git log с аргументом –p и -2

```
C:\Users\AND\laba2>git log -p -2
commit 66f72012647f658ae9a22186e2c970df36bbe9b2 (HEAD -> main, origin/main, origin/HEAD)
Author: midand-hash <midovandik07@gmail.com>
Date:   Thu Dec 26 17:03:53 2024 +0300

    Внёс изменения для точности кода

diff --git a/pythone.py b/pythone.py
index aaf2d91..958c1b4 100644
--- a/pythone.py
+++ b/pythone.py
@@ -1,11 +1,21 @@
 # Программа для расчета площади прямоугольника

 # Запрашиваем длину и ширину прямоугольника
-length = float(input("Введите длину прямоугольника: "))
-width = float(input("Введите ширину прямоугольника: "))
+try:
+    length = float(input("Введите длину прямоугольника: "))
+    width = float(input("Введите ширину прямоугольника: "))
+except ValueError:
+    print("Ошибка: Введите числовые значения для длины и ширины.")
+    exit()
+
+# Проверка на положительность
+if length <= 0 or width <= 0:
+    print("Длина и ширина должны быть положительными числами.")
+    exit()

 # Вычисляем площадь
 area = length * width
```

Рисунок 3. Аргумент –p -2 команды git log

4. Использование опции --stat команды git log

```
C:\Users\AND\laba2>git log --stat
commit 66f72012647f658ae9a22186e2c970df36bbe9b2 (HEAD -> main, origin/main, origin/HEAD)
Author: midand-hash <midovandik07@gmail.com>
Date:   Thu Dec 26 17:03:53 2024 +0300

    Внёс изменения для точности кода

 pythone.py | 14 ++++++++++++--
 1 file changed, 12 insertions(+), 2 deletions(-)

commit 96c0441e2ef410bada2cef6781b24af72d3700b7
Author: midand-hash <midovandik07@gmail.com>
Date:   Thu Dec 26 17:00:03 2024 +0300

    Написал код в новом файле

 pythone.py | 11 ++++++++++
 1 file changed, 11 insertions(+)

commit d7fbf637059846d8e713918404b6c126e91172fa
Author: midand-hash <midovandik07@gmail.com>
Date:   Thu Dec 26 16:55:24 2024 +0300

    Добавил новый пустой файл

 pythone.py | 0
 1 file changed, 0 insertions(+), 0 deletions(-)

commit 49d318f94736b94c7c4d30e997c357b4632e2bf6
Author: midand-hash <midovandik07@gmail.com>
Date:   Thu Dec 26 16:52:31 2024 +0300

    Добавил информацию в README

 README.md | 4 +++-
 1 file changed, 3 insertions(+), 1 deletion(-)

commit 6092dead549482ba30be360549bfc37459be9b7f
Author: midand-hash <midovandik07@gmail.com>
Date:   Thu Dec 26 16:42:12 2024 +0300

    Initial commit

 .gitignore | 171 ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
 LICENSE    |  21 ++++++++
 README.md  |   1 +
 3 files changed, 193 insertions(+)
```

Рисунок 4. Опция git log –stat

5. Изменение формата вывода с помощью опции -pretty=oneline

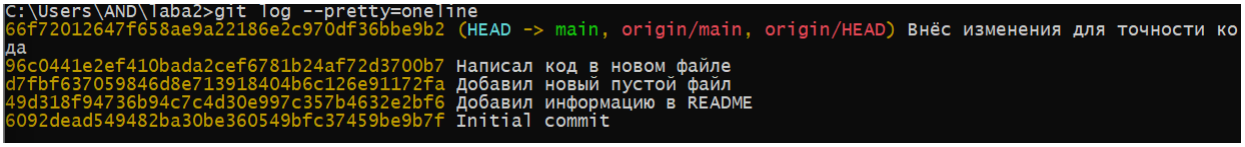


Рисунок 5. Вывод с опцией --pretty=oneline

6. Использование форматного вывода опции --pretty

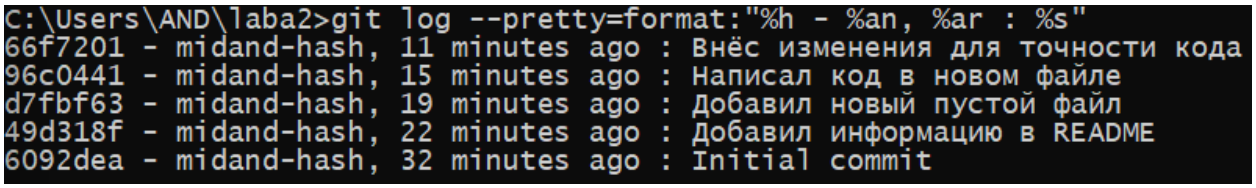


Рисунок 6. Форматный вывод

7. Использование опции -pretty с опцией --graph

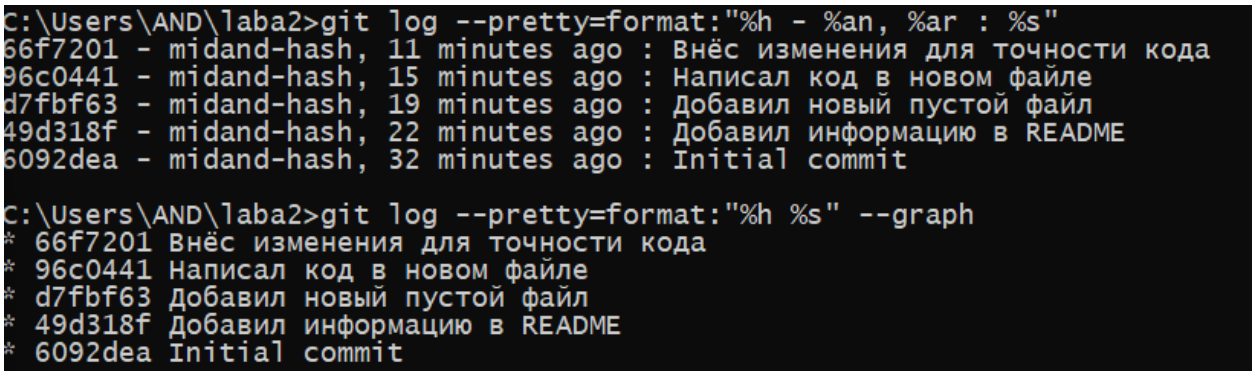


Рисунок 7. Вывод с опцией --graph

8. Создание тега


Рисунок 8. Создание тега

9. Просмотр журнала хранилища

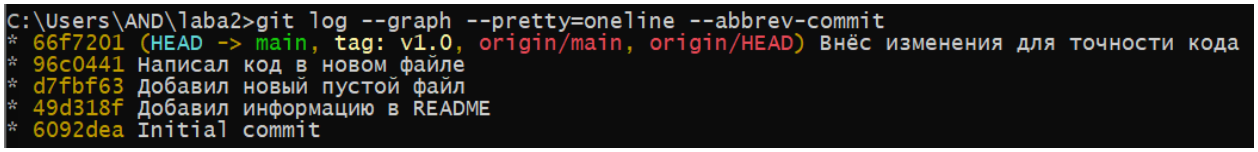


Рисунок 9. Журнал хранилища

10. Просмотр содержимого коммита с указанным хэшем

```
C:\Users\AND\laba2>git show 96c0441
commit 96c0441e2ef410bada2cef6781b24af72d3700b7
Author: midand-hash <midovandik07@gmail.com>
Date:   Thu Dec 26 17:00:03 2024 +0300

    Написал код в новом файле

diff --git a/pythone.py b/pythone.py
index e69de29..aaf2d91 100644
--- a/pythone.py
+++ b/pythone.py
@@ -0,0 +1,11 @@
+# Программа для расчета площади прямоугольника
+
+# Запрашиваем длину и ширину прямоугольника
+length = float(input("Введите длину прямоугольника: "))
+width = float(input("Введите ширину прямоугольника: "))
+
+# Вычисляем площадь
+area = length * width
+
+# Выводим результат
+print("Площадь прямоугольника:", area)
```

Рисунок 10. Содержимое коммита 96c0441

11. Просмотр содержимого предпоследнего коммита

```
C:\Users\AND\laba2>git show HEAD~1
commit 96c0441e2ef410bada2cef6781b24af72d3700b7
Author: midand-hash <midovandik07@gmail.com>
Date:   Thu Dec 26 17:00:03 2024 +0300

    Написал код в новом файле

diff --git a/pythone.py b/pythone.py
index e69de29..aaf2d91 100644
--- a/pythone.py
+++ b/pythone.py
@@ -0,0 +1,11 @@
+# Программа для расчета площади прямоугольника
+
+# Запрашиваем длину и ширину прямоугольника
+length = float(input("Введите длину прямоугольника: "))
+width = float(input("Введите ширину прямоугольника: "))
+
+# Вычисляем площадь
+area = length * width
+
+# Выводим результат
+print("Площадь прямоугольника:", area)
```

Рисунок 11. Содержимое предпоследнего коммита

12. Удаление кода программы и возврат изменений командой git checkout

```
C:\Users\AND\laba2>git status
On branch main
Your branch is up to date with 'origin/main'.

Changes not staged for commit:
  (use "git add/rm <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        deleted:    pythone.py

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

Рисунок 12. Удаление программы

13. Возврат изменении.

```
C:\Users\AND\laba2>git checkout -- pythone.py

C:\Users\AND\laba2>git status
On branch main
Your branch is up to date with 'origin/main'.

nothing to commit, working tree clean
```

Рисунок 13. Возврат изменении с checkout

14. Создание коммита и откат к предыдущей версии командой git reset

```
C:\Users\AND\laba2>git reset --hard HEAD~1
HEAD is now at 96c0441 Написал код в новом файле
```

Рисунок 14. Откат коммита

Ссылка на репозитории: https://github.com/midand-hash/laba2.git

**Ответы на контрольные вопросы:**

1. Что такое СКВ и каково ее назначение?

Система контроля версий (СКВ) — это система, регистрирующая изменения в одном или нескольких файлах с тем, чтобы в дальнейшем была возможность вернуться к определённым старым версиям этих файлов.

2. В чем недостатки локальных и централизованных СКВ?

Локальные СКВ (системы контроля версий) менее удобны для совместной работы, поскольку требуют постоянной синхронизации

изменений между участниками проекта. Централизованные СКВ, в свою очередь, могут быть уязвимы к отказам сервера, а также к потере данных, если главный репозиторий поврежден.

3. К какой СКВ относится Git?

Git относится к децентрализованным системам контроля версий.

4. В чем концептуальное отличие Git от других СКВ?

Git отличается от других СКВ тем, что он децентрализован: каждый разработчик имеет полную копию репозитория, а не только доступ к нему. Это позволяет работать автономно, делать коммиты локально и синхронизировать изменения с другими разработчиками.

5. Как обеспечивается целостность хранимых данных в Git?

Git использует криптографическую хеш-функцию SHA-1 для вычисления уникального хэша для каждого файла и коммита. Изменения в файле изменяют его хэш, что позволяет Git отслеживать изменения и обеспечивать целостность данных.

6. В каких состояниях могут находится файлы в Git? Как связаны эти состояния? Файлы в Git могут находиться в 3 состояниях:

– Untracked: файл не отслеживается Git, т.е. не включен в репозиторий.

– Staged: файл был изменен и подготовлен к коммиту, т.е. включен в следующий коммит.

– Committed: файл был зафиксирован в истории репозитория.

– Связь между состояниями:

1. Untracked -> Staged: файл переходит в состояние Staged после команды git add.

2. Staged -> Committed: файл переходит в состояние Committed после команды git commit.

7. Что такое профиль пользователя в GitHub?

Профиль пользователя на GitHub — это публичная страница, которая представляет вас как разработчика. На ней отображаются ваши репозитории,

активности, подписки, а также информация о вас: имя, местоположение, сайт, био, аватар.

8. Какие бывают репозитории в GitHub?

– Публичные: доступны всем пользователям GitHub. Их код можно просматривать, скачивать и использовать бесплатно.

– Приватные: доступны только владельцу репозитория и приглашенным сотрудникам. Для приватных репозиториев нужна платная подписка GitHub.

– Также существуют fork-репозитории: это копии публичных репозиториев, которые можно модифицировать и использовать для собственных проектов. Fork-репозитории могут быть как публичными, так и приватными.

9. Укажите основные этапы модели работы с GitHub.

Основные этапы работы с GitHub:

1. Создание репозитория: Создается новый репозиторий на GitHub, где будет храниться код проекта.

2. Клонирование репозитория: Локальная копия репозитория скачивается на компьютер с помощью команды git clone.

3. Работа с кодом: В локальной копии вносятся изменения в код проекта.

4. Добавление изменений в индекс: Измененные файлы добавляются в индекс с помощью команды git add.

5. Коммит: Изменения в индексе сохраняются в локальной истории с помощью команды git commit.

6. Пуш: Изменения из локального репозитория отправляются на сервер GitHub с помощью команды git push.

7. Pull: Изменения из репозитория GitHub скачиваются в локальную копию с помощью команды git pull.

10. Как осуществляется первоначальная настройка Git после установки?

git config - это команда, которая позволяет вам настроить Git под свой стиль работы и предпочтения. Она используется для установки глобальных, локальных и системных параметров Git. Глобальные параметры: применяются ко всем репозиториям на вашем компьютере. Их установка осуществляется с флагом --global. Локальные параметры: применяются только к текущему репозиторию. Их установка осуществляется без флага --global. Системные параметры: применяются ко всем пользователям системы. Их установка осуществляется с флагом --system.

11. Опишите этапы создания репозитория в GitHub.

1. Авторизоваться на сайте.

2. Нажать кнопку "New" и ввести название, описание и тип доступа для нового репозитория.

3. Нажать кнопку "Create repository".

12. Какие типы лицензий поддерживаются GitHub при создании репозитория?

– MIT: Позволяет свободное использование, модификацию и распространение, включая коммерческие цели.

– Apache 2.0: Позволяет свободное использование, модификацию и распространение, включая коммерческие цели, с условием сохранения исходной лицензии.

– GPL 3.0: Позволяет свободное использование, модификацию и распространение, включая коммерческие цели, но с обязательством публикации исходного кода под той же лицензией.

– BSD 3-Clause: Позволяет свободное использование, модификацию и распространение, включая коммерческие цели, с условием сохранения имени автора и отказа от ответственности за использование программного обеспечения.

– Public Domain: Отказывается от всех прав на программное обеспечение, позволяя свободное использование, модификацию и

распространение, без каких-либо ограничений.

– Также GitHub предоставляет возможность выбрать "None" (без лицензии), что означает, что вы не предоставляете никаких прав на использование вашего кода.

13. Как осуществляется клонирование репозитория GitHub?

Зачем нужно клонировать репозиторий? Клонирование репозитория GitHub осуществляется с помощью команды git clone: git clone <URL_репозитория> Клонирование репозитория создает локальную копию репозитория на вашем компьютере, что позволяет вам: Работать с кодом локально: Вносить изменения, создавать ветки, использовать Git без подключения к Интернету. Вносить вклад в проект: Вносить изменения в код проекта и отправлять их на GitHub. Сохранять историю изменений: Локальная копия сохраняет полную историю всех изменений в репозитории. Совместная работа: Клонирование позволяет нескольким разработчикам работать над проектом одновременно и синхронизировать свои изменения.

14. Как проверить состояние локального репозитория Git?

Чтобы проверить состояние локального репозитория Git, используется команда git status. Она покажет:

• Измененные файлы: Файлы, которые были изменены, но еще не добавлены в индекс.

• Файлы, добавленные в индекс: Файлы, которые были изменены и добавлены в индекс для следующего коммита.

• Неотслеживаемые файлы: Файлы, которые не отслеживаются Git (не входят в репозиторий).

• Текущая ветка: Ветка, в которой вы находитесь.

• Состояние ветки: Информирует о том, соответствует ли локальная копия ветки удаленной копии на GitHub.

15. Как изменяется состояние локального репозитория Git после выполнения следующих операций: добавления/изменения файла в локальный репозиторий Git; добавления нового/ измененного файла под версионный

контроль с помощью команды git add ; фиксации (коммита) изменений с помощью команды git commit и отправки изменений на сервер с помощью команды git push ? После добавления или изменения файла в локальном репозитории Git он переходит в состояние "Untracked" (не отслеживается). Git еще не знает об этом файле и не включает его в историю изменений. Команда git add <имя_файла> добавляет файл в индекс Git. Файл переходит в состояние "Staged" (подготовлен к коммиту). Теперь Git знает об этом файле и будет включать его в следующий коммит. Команда git commit -m "Сообщение о коммите" создает новый коммит в локальной истории Git. Файл переходит в состояние "Committed" (зафиксирован). Изменения, которые были в индексе, теперь стали частью истории репозитория.

Команда git push origin main отправляет изменения из локальной

истории на сервер GitHub. Локальные коммиты теперь доступны в удаленном

репозитории.

16. У Вас имеется репозиторий на GitHub и два рабочих компьютера, с помощью которых Вы можете осуществлять работу над некоторым проектом с использованием этого репозитория. Опишите последовательность команд, с помощью которых оба локальных репозитория, связанных с репозиторием GitHub будут находиться в синхронизированном состоянии. Примечание: описание необходимо начать с команды git clone.

1. Клонировать репозиторий на оба компьютера: git clone <URL_репозитория>.
2. Работать над проектом на первом компьютере, фиксировать изменения (git add, git commit).
3. Отправить изменения на GitHub: git push origin main.
4. Обновить локальный репозиторий на втором компьютере: git pull origin main.
5. Повторить шаги 2-4 для второго компьютера.

17. GitHub является не единственным сервисом, работающим с Git. Какие сервисы еще Вам известны? Приведите сравнительный анализ одного из таких сервисов с GitHub. Bitbucket: Разработан Atlassian, известен своей интеграцией с другими продуктами Atlassian, такими как Jira и Confluence. Предлагает как бесплатные, так и платные планы с поддержкой приватных репозиториев.

- GitLab: Открытый сервис с возможностью развертывания на собственных серверах. Предлагает множество функций для управления кодом, CI/CD, и безопасности.
- Azure DevOps: Облачный сервис от Microsoft, включающий в себя систему контроля версий Git, CI/CD, управление проектами и другие инструменты.

GitHub и Bitbucket - популярные платформы для хостинга Git репозиториев, но у них есть некоторые различия. GitHub предлагает бесплатные публичные репозитории и платные приватные, в то время как Bitbucket предоставляет бесплатные приватные репозитории для небольших команд и платные планы с расширенными функциями. GitHub интегрируется с различными сервисами, такими как Travis CI и CircleCI, в то время как Bitbucket отлично интегрируется с продуктами Atlassian, такими как Jira и Confluence. GitHub предлагает более широкий набор функций для управления кодом, сотрудничества и развертывания, в то время как Bitbucket предлагает набор функций, достаточный для большинства проектов. Сообщество разработчиков на GitHub значительно больше, чем на Bitbucket, но Bitbucket обеспечивает хорошую поддержку Atlassian.

18. Интерфейс командной строки является не единственным и далеко не самым удобным способом работы с Git. Какие Вам известны программные средства с графическим интерфейсом пользователя для работы с Git? Приведите как реализуются описанные в лабораторной работе операции Git с помощью одного из таких программных средств. GitHub Desktop: Разработан

самой компанией GitHub, интегрирован с веб-платформой GitHub и предлагает простой и интуитивно понятный интерфейс.

• GitKraken: Известен своим стильным дизайном, большим количеством функций и поддержкой многих систем контроля версий, включая Git, Mercurial и SVN.

• Sourcetree: Разработан Atlassian, производителем Jira и Bitbucket, имеет хорошую интеграцию с этими сервисами и предлагает широкий набор функций для управления репозиториями.

• TortoiseGit: Бесплатная программа с открытым исходным кодом для Windows, известна своей интеграцией в контекстное меню Проводника Windows и простой использованием. Как реализуются операции Git в GitHub Desktop В GitHub Desktop необходимо нажать кнопку "Clone a repository" и ввести URL репозитория. После чего выбрать папку для клонирования.

**Вывод:** в ходе лабораторной работы были исследованы базовые возможности системы контроля версий Git и веб-сервиса для хостинга ITпроектов GitHub.