

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:
Костенко Савелий Дмитриевич
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:
Доцент департамента цифровых,
робототехнических систем и
электроники Воронкин Р.А

(подпись)

Отчет защищен с оценкой _____ Дата защиты _____

Ставрополь, 2024 г.

Тема: Исследование возможностей Git для работы с локальными репозиториями

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

Порядок выполнения работы:

1. Были изучены основные теоретические сведения.
2. Был создан общедоступный репозиторий:

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Required fields are marked with an asterisk ().*

Owner * **Repository name ***

Gbeek / 2 laba

✓ Your new repository will be created as 2-laba.
The repository name can only contain ASCII letters, digits, and the characters ., -, and _.

Great repository names are short and memorable. Need inspiration? How about **super-duper-fiesta** ?

Description (optional)

☒ **Public**
Anyone on the internet can see this repository. You choose who can commit.

☐ **Private**
You choose who can see and commit to this repository.

Initialize this repository with:

☒ **Add a README file**
This is where you can write a long description for your project. [Learn more about READMEs.](#)

Add .gitignore

.gitignore template: **Python**

Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)

Choose a license

License: **MIT License**

A license tells others what they can and can't do with your code. [Learn more about licenses.](#)

This will set **main** as the default branch. Change the default name in your [settings](#).

You are creating a public repository in your personal account.

Create repository

Рисунок 1. Создание общедоступного репозитория

3. Был скопирован репозиторий на компьютер:

```
C:\Users\savel>git clone https://github.com/Gbeek/lab2.git
Cloning into 'lab2'...
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 (from 0)
Receiving objects: 100% (5/5), done.
```

Рисунок 2. Клонирование репозитория

4. Была добавлена в файл README.md дополнительная информация:

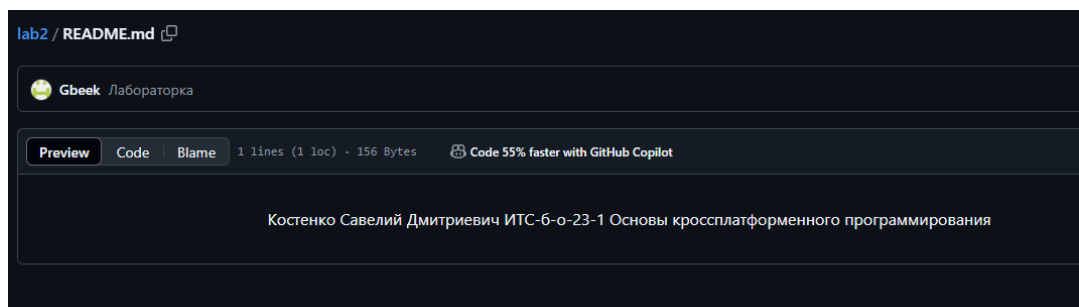


Рисунок 3. Информацию в файле README

5. Была написана программа на Java и зафиксированы изменения:

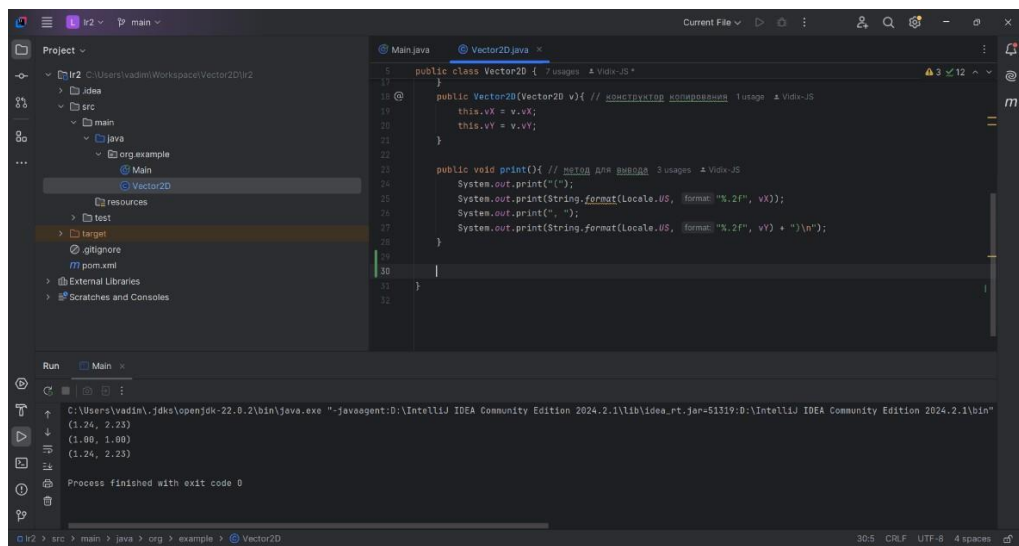


Рисунок 4. Программа, написанная на Java

```
C:\Users\vadim>cd c:\users\vadim\workspace\vector2d
c:\Users\vadim\Workspace\Vector2D>git add .
c:\Users\vadim\Workspace\Vector2D>git status
On branch main
Your branch is up to date with 'origin/main'.

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

c:\Users\vadim\Workspace\Vector2D>git commit -m "Изменение файла README.md"
[main 2a36067] Изменение файла README.md
 1 file changed, 4 insertions(+)

c:\Users\vadim\Workspace\Vector2D>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), 471 bytes | 235.00 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/Vidix-JS/Vector2D.git
 4872aa6..2a36067  main -> main
c:\Users\vadim\Workspace\Vector2D>
```

Рисунок 5. Фиксирование изменений программы

6. Были добавили теги:

```

c:\Users\vadim\Workspace\Vector2D>git commit -m "Добавление конструктора по умолчанию, конструктора при двух параметрах, конструктора копирования и метод для вывода вектора и использование их в методе main"
[main 07b9a01] Добавление конструктора по умолчанию, конструктора при двух параметрах, конструктора копирования и метод для вывода вектора и использование их в методе main
2 files changed, 30 insertions(+), 1 deletion(-)

c:\Users\vadim\Workspace\Vector2D>git tag -a v1.0 "Базовые функции при работе с векторами"
fatal: Failed to resolve 'Базовые функции при работе с векторами' as a valid ref.

c:\Users\vadim\Workspace\Vector2D>git tag -a v1.0 "version 1.0"
fatal: Failed to resolve 'version 1.0' as a valid ref.

c:\Users\vadim\Workspace\Vector2D>git tag -a v1.0 -m "Базовые функции при работе с векторами"

c:\Users\vadim\Workspace\Vector2D>git push
Enumerating objects: 19, done.
Counting objects: 100% (19/19), done.
Delta compression using up to 8 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (10/10), 1.20 KiB | 409.00 KiB/s, done.
Total 10 (delta 2), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To https://github.com/Vidix-JS/Vector2D.git
 9d6ee93..07b9a01  main -> main

c:\Users\vadim\Workspace\Vector2D>git push origin v1.0
Enumerating objects: 1, done.
Counting objects: 100% (1/1), done.
Writing objects: 100% (1/1), 227 bytes | 227.00 KiB/s, done.
Total 1 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/Vidix-JS/Vector2D.git
  * [new tag]          v1.0 -> v1.0

```

Рисунок 6. Добавление тега

7. Были созданы не менее 7 коммитов:

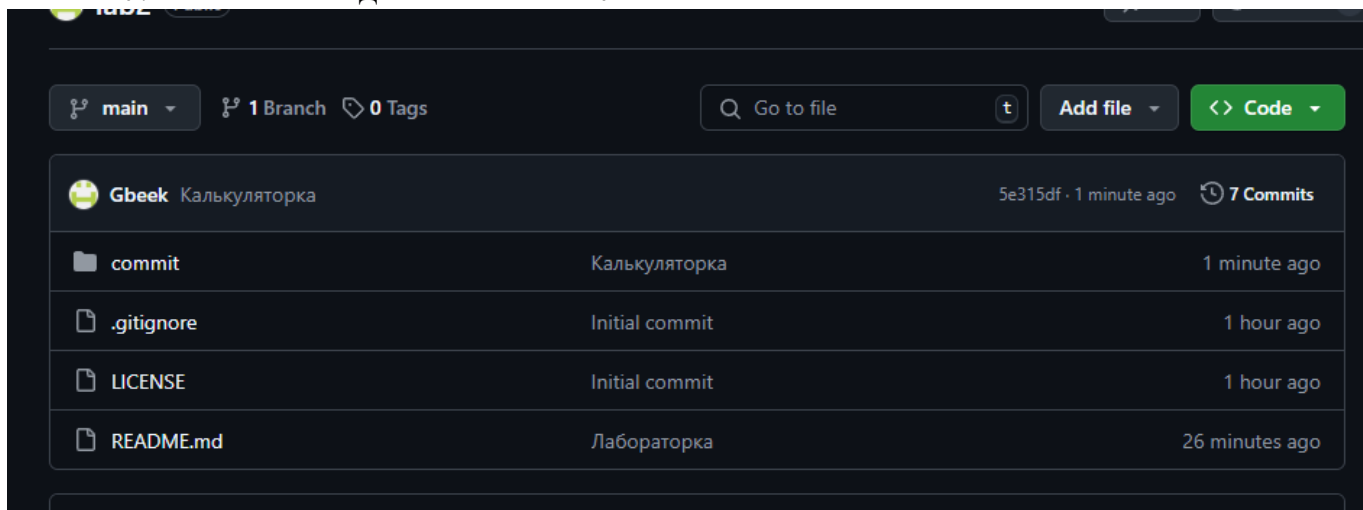


Рисунок 7. Созданы коммиты

8. Была просмотрена история коммитов с помощью команды git log

```

C:\Users\savel\lab2>git log
commit 5e315df7cb85eb1f17cbd5c1ec51ba37d7d2a6e6 (HEAD -> main, origin/main, origin/HEAD)
Author: Gbeek <Gbeek337@gmail.com>
Date: Sun Dec 29 19:56:24 2024 +0300

    Калькуляторка

commit a8bf7c9ee69fc6e1f409590e257fae9dae07692a
Author: Gbeek <Gbeek337@gmail.com>
Date: Sun Dec 29 19:51:41 2024 +0300

    Улучш кальк

commit 67a2d1d7bfad3e5d19eedb02ac5cafbbfe049e4b1
Author: Gbeek <Gbeek337@gmail.com>
Date: Sun Dec 29 19:50:12 2024 +0300

    Начальный калькулятор

commit b0765a7db699946e620f415c24d568e4bca1aabb
Author: Gbeek <Gbeek337@gmail.com>
Date: Sun Dec 29 19:46:36 2024 +0300

    Начало кода

```

Рисунок 8. Просмотр истории коммитов

9. Просмотр последнего коммита с помощью команды git show

HEAD:

```
Калькуляторка
diff --git a/commit/calc ssssss.py b/commit/calc ssssss.py
new file mode 100644
index 0000000..56d62ee
--- /dev/null
+++ b/commit/calc ssssss.py
@@ -0,0 +1,52 @@
+# Функция сложения
+def add(x, y):
+    return x + y
+
+# Функция вычитания
+def subtract(x, y):
+    return x - y
+
+# Функция умножения
+def multiply(x, y):
+    return x * y
+
+# Функция деления
+def divide(x, y):
+    if y == 0:
+        return "Ошибка: деление на ноль"
+    return x / y
+
+# Функция для выполнения вычислений
+def calculate(num1, operation, num2):
+    if operation == '+':
+        return add(num1, num2)
+    elif operation == '-':
+        return subtract(num1, num2)
+    elif operation == '*':
+        return multiply(num1, num2)
+    elif operation == '/':
+        return divide(num1, num2)
+    else:
+        return "Неверная операция"
+
+if __name__ == "__main__":
+    print("Простой калькулятор")
+
+    while True:
+        try:
+            num1 = float(input("Введите первое число: "))
+            operation = input("Введите операцию (+, -, *, /) (или 'q' для выхода): ")
+
+            if operation == 'q':
+                break
+
+            num2 = float(input("Введите второе число: "))
+            result = calculate(num1, operation, num2)
+            print(f"Результат: {result}")
+
+        except ValueError:
+            print("Введите корректные значения")
+
+        except KeyboardInterrupt:
+            print("Выход из программы")
+            break
```

Рисунок 9. Последний коммит

Вывод: в результате использования команды `git reset --hard` отменяется последние коммиты и сбрасывается рабочая копия до определенного состояния.

Ответы на контрольные вопросы:

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

Для просмотра истории коммитов в Git используется команда `git log`. Она отображает список коммитов в хронологическом порядке, начиная с самого последнего.

При помощи аргумента `-p` можно посмотреть разницу внесенную в каждый коммит.

При помощи параметра `-n`, где `n` это число, можно посмотреть историю последних `n`-коммитов.

2. Как ограничить вывод при просмотре истории коммитов?

Для ограничения вывода при просмотре истории коммитов в Git можно использовать несколько опций:

- Ограничение количества коммитов; `-(n)`;
- Фильтрация по времени;

С помощью `-since`, `-after` показывает те коммиты, которые сделаны после указанной даты

С помощью `--until`, `--before` показывает только те коммиты, которые были сделаны до указанной даты.

- Фильтрация по автору;
`--author`
- Фильтрация по сообщению коммита;
`--committer`
- Ограничение вывода по формату.
`--pretty=format"`

3. Как внести изменения в уже сделанный коммит?

- `git commit --amend`;
- `git reset <имя_коммита>`.

4. Как отменить индексацию файла в Git?

`git checkout`. Эта команда отменит все изменения, внесенные в файлы, вернув их к состоянию в последнем коммите.

`git reset HEAD`. Эта команда удалит файл из индекса, но оставит его в рабочей области.

5. Как отменить изменения в файле?

В Git есть несколько способов отменить изменения в файле:

- `git checkout -- <имя файла>`;
- `git reset HEAD <имя файла>`;
- `git restore <имя файла>`;
- `git revert <имя файла>`;
- `git stash`

6. Что такое удаленный репозиторий Git?

Удаленный репозиторий Git – это хранилище кода, доступное с разных компьютеров через сеть.

7. Как выполнить просмотр удаленных репозиториях данного локального репозитория?

Для просмотра удаленных репозиториях, связанных с вашим локальным репозиторием Git, используйте команду `git remote`. Эта команда выведет

список имен, удаленных репозиторий, которые вы добавили к своему локальному репозиторию.

8. Как добавить удаленный репозиторий для данного локального репозитория?

Чтобы добавить удаленный репозиторий к вашему локальному репозиторию Git, используйте команду `git remote add`: `git remote add <имя_удаленного_репозитория> <URL_удаленного_репозитория>`

9. Как выполнить отправку/получение изменений с удаленного репозитория?

Отправка изменений (push): `git push <имя_удаленного_репозитория><ветвь>`: Эта команда отправляет ваши локальные изменения в указанную ветвь на удаленный репозиторий.

Получение изменений (pull): `git pull <имя_удаленного_репозитория><ветвь>`: Эта команда получает изменения из указанной ветви на удаленном репозитории и объединяет их с вашей локальной веткой

10. Как выполнить просмотр удаленного репозитория?

- С помощью команды `git log`;
- С помощью команды `git fetch`;
- С помощью команды `git remote show`.

11. Каково назначение тэгов Git?

Тэги Git – это метки, которые позволяют пометить определенные коммиты в репозитории. Они служат для идентификации важных моментов в истории проекта, таких как релизы, версии или отправные точки для ветвления.

12. Как осуществляется работа с тэгами Git?

- Создание тега;
- Просмотр тегов;
- Перемещение тега;
- Удаление тега;

- Отправка тегов на удаленный репозиторий;
- Просмотр информации о теге;
- Переключение на коммит, помеченный тегом.

13. Самостоятельно изучите назначение флага `--prune` в командах `git fetch` и `git push`. Каково назначение этого флага?

Флаг `--prune` в командах `git fetch` и `git push` используется для удаления удаленных веток, которые уже не существуют на удаленном сервере.

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

Ссылка на GitHub: <https://github.com/Vidix-JS/Vector2D>