

**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 №1.1**

**Дисциплина: «Программирование на Python»**

**Тема: «Исследование возможностей повторного использования кода в  
структурном программировании»**

**Выполнил: студент 2 курса**

**группы ИВТ-б-о-22-1**

**Болуров Ислам Расулович**

**Ставрополь 2023**

Выполнение работы:

## 1. Создание репозитория и его настройка.

### 1.1 Настроить как требуется в задании и создать репозиторий в GitHub:


Owner \*  / Repository name \*


✔ Your new repository will be created as Lab-1.1.  
The repository name can only contain ASCII letters, digits, and the characters ., -, and \_.

Great repository names are short and memorable. Need inspiration? How about [laughing-garbanzo](#) ?

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


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

Choose a 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 репозитория

1.2 Добавить информацию в README.md и добавить необходимые правила для языка программирования C++ и среды разработки VisualStudioCode в .gitignore:

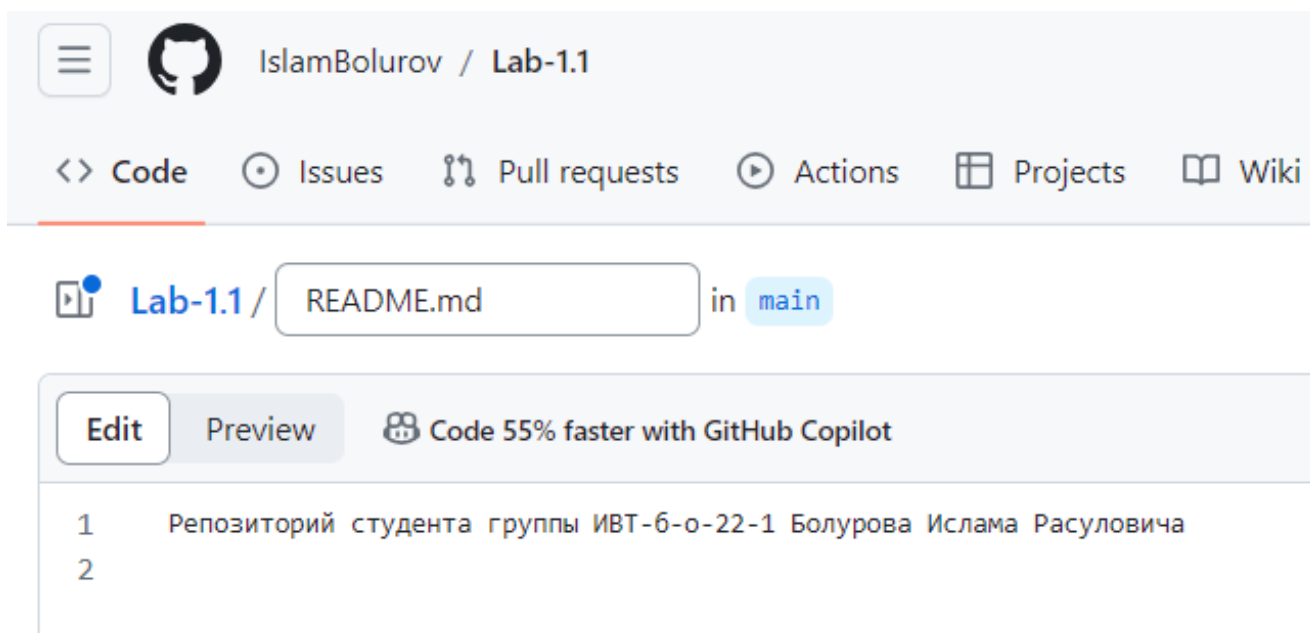


Рисунок 1.2 Изменения в README.md

## Commit

### Update .gitignore

main

IslamBolurov committed 37 minutes ago Verified

Showing 1 changed file with 13 additions and 0 deletions.

13 .gitignore		
↑...	@@ -30,3 +30,16 @@	
30	30	*.exe
31	31	*.out
32	32	*.app
33	+	
34	+ .vscode/*	
35	+ !.vscode/settings.json	
36	+ !.vscode/tasks.json	
37	+ !.vscode/launch.json	
38	+ !.vscode/extensions.json	
39	+ !.vscode/*.code-snippets	
40	+	
41	+ # Local History for Visual Studio Code	
42	+ .history/	
43	+	
44	+ # Built Visual Studio Code Extensions	
45	+ *.vsix	

Рисунок 1.3 Изменения в .gitignore

## 2. Клонирование репозитория на локальное хранилище, создание и PUSH программы.

### 2.1 Клонировать репозиторий на свой ПК:

```
IslamBolurov@Islam666 MINGW64 ~/Desktop/gitgitgit
$ git clone https://github.com/IslamBolurov/Lab-1.1.git
Cloning into 'Lab-1.1'...
remote: Enumerating objects: 11, done.
remote: Counting objects: 100% (11/11), done.
remote: Compressing objects: 100% (10/10), done.
remote: Total 11 (delta 2), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (11/11), done.
Resolving deltas: 100% (2/2), done.
```

Рисунок 2.1 Клонирование репозитория на ПК

### 2.2 Создать программу на C++ в репозитории:

от компьютер > Рабочий стол > gitgitgit > Lab-1.1 >

Имя	Дата изменения	Тип	Размер
programm	20.11.2023 9:26	Папка с файлами	
.gitignore	20.11.2023 9:20	Текстовый докум...	1 КБ
LICENSE	20.11.2023 9:20	Файл	2 КБ
README	20.11.2023 9:20	Исходный файл ...	1 КБ

Рисунок 2.1 Папка с проектом C++

```
program.cpp X
program.cpp
1  #include <iostream>
2
3  int main()
4  {
5      std::cout<<"Hello World\n";
6  }
```

Рисунок 2.2 Код программы

### 2.2 Сделал коммит созданной программы и PUSH на удаленный сервер:

```

IslamBolurov@Islam666 MINGW64 ~/Desktop/gitgitgit/Lab-1.1 (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

Untracked files:
  (use "git add <file>..." to include in what will be committed)
  programm/

nothing added to commit but untracked files present (use "git add" to track)

IslamBolurov@Islam666 MINGW64 ~/Desktop/gitgitgit/Lab-1.1 (main)
$ git add programm

IslamBolurov@Islam666 MINGW64 ~/Desktop/gitgitgit/Lab-1.1 (main)
$ 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)
  new file:   programm/programm.cpp

IslamBolurov@Islam666 MINGW64 ~/Desktop/gitgitgit/Lab-1.1 (main)
$ git commit -m "programm Hello World"
[main 41543a9] programm Hello World
1 file changed, 5 insertions(+)
create mode 100644 programm/programm.cpp

IslamBolurov@Islam666 MINGW64 ~/Desktop/gitgitgit/Lab-1.1 (main)
$ git status
On branch main
Your branch is ahead of 'origin/main' by 1 commit.
(use "git push" to publish your local commits)

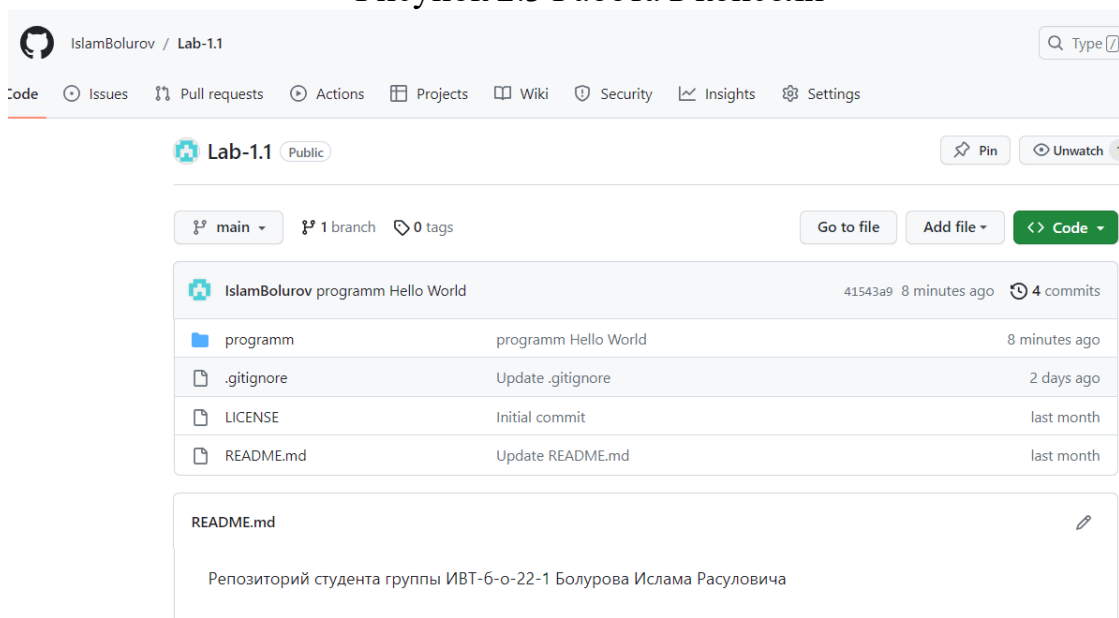
nothing to commit, working tree clean

IslamBolurov@Islam666 MINGW64 ~/Desktop/gitgitgit/Lab-1.1 (main)
$ git push
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 4 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (4/4), 389 bytes | 194.00 KiB/s, done.
Total 4 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/IslamBolurov/Lab-1.1.git
75e7d1f..41543a9  main -> main

IslamBolurov@Islam666 MINGW64 ~/Desktop/gitgitgit/Lab-1.1 (main)
$ |

```

Рисунок 2.3 Работа в консоли



IslamBolurov / Lab-1.1

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

Lab-1.1 Public Pin Unwatch 1

main 1 branch 0 tags Go to file Add file <> Code

File	Commit	Time
programm	programm Hello World	8 minutes ago
.gitignore	Update .gitignore	2 days ago
LICENSE	Initial commit	last month
README.md	Update README.md	last month

README.md

Репозиторий студента группы ИБТ-6-о-22-1 Болурова Ислама Расуловича

Рисунок 2.4 Изменения на удаленном репозитории

**3. Зафиксировать изменения при написании программы на локальном репозитории и запустить на удаленный.**

```

1  #include <iostream>
2  using namespace std;
3  int main()
4  {
5  cout<<"Hello World\n";
6  }

```

Рисунок 3.1 Изменение программы

```

IslamBolurov@Islam666 MINGW64 ~/Desktop/gitgitgit/Lab-1.1 (main)
$ git commit -m "5 com"
[main 5cd635d] 5 com
1 file changed, 2 insertions(+), 1 deletion(-)

```

Рисунок 3.2 5-й коммит

```

1  #include <iostream>
2  using namespace std;
3  int main()
4  {
5  cout<<"Hello World\n";
6  cout<<1+1;
7  }

```

Рисунок 3.3 Изменение программы

```

IslamBolurov@Islam666 MINGW64 ~/Desktop/gitgitgit/Lab-1.1 (main)
$ git commit -m "6 com"
[main 32a3e6f] 6 com
1 file changed, 1 insertion(+)

```

Рисунок 3.4 6-й коммит

```

1  #include <iostream>
2  using namespace std;
3  int main()
4  {
5  cout<<"Hello World\n";
6  cout<<1+1;
7  cout<<"\nIslamBolurov";
8  }

```

Рисунок 3.5 Изменение программы

```

IslamBolurov@Islam666 MINGW64 ~/Desktop/gitgitgit/Lab-1.1 (main)
$ git commit -m "7 com"
[main 9edb4f7] 7 com
1 file changed, 1 insertion(+)

```

Рисунок 3.6 7-й коммит

Сделать push репозитория на удаленный репозиторий:

```
IslamBolurov@Islam666 MINGW64 ~/Desktop/gitgitgit/Lab-1.1 (main)
$ git push
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 4 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (4/4), 352 bytes | 176.00 KiB/s, done.
Total 4 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To https://github.com/IslamBolurov/Lab-1.1.git
 32a3e6f..9edb4f7  main -> main
```

Рисунок 3.7 Команда push в консоли

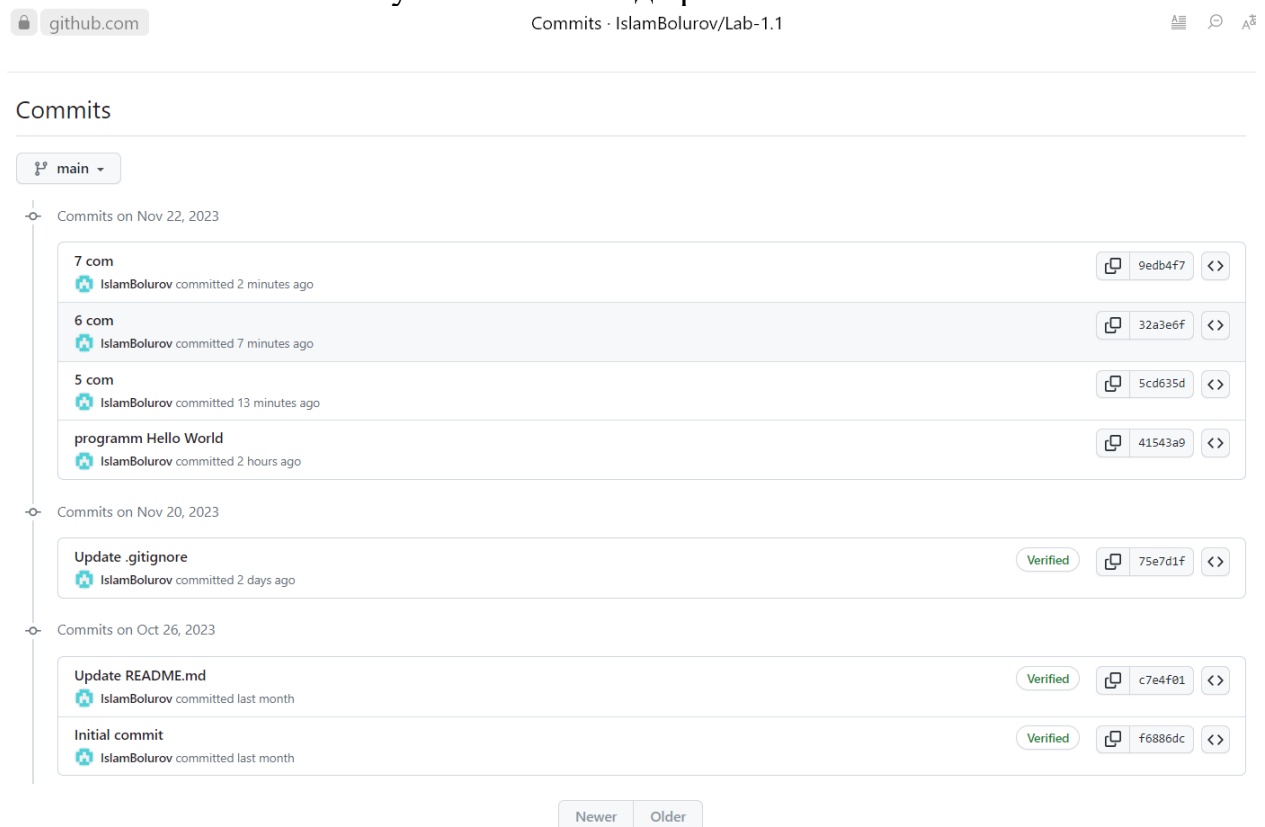


Рисунок 3.8 Список коммитов в github-е

## 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: многие в качестве метода контроля версий применяют копирование файлов в отдельную директорию. Такой подход очень распространён из-за его простоты, однако он невероятно сильно подвержен появлению ошибок. Можно легко забыть, в какой директории находитесь, и случайно изменить не тот файл или скопировать не те файлы, которые вы хотели.

Централизованные СКВ: единая точка отказа, представленная централизованным сервером. Если этот сервер выйдет из строя на час, то в течение этого времени никто не сможет использовать контроль версий для сохранения изменений, над которыми работает, а также никто не сможет обмениваться этими изменениями с другими разработчиками.

### **3. К какой СКВ относится Git?**

Git относится к распределённым системам, поэтому не зависит от центрального сервера, где хранятся файлы.

### **4. В чем концептуальное отличие Git от других СКВ?**

Git не хранит и не обрабатывает данные таким же способом как другие СКВ. Каждый раз, когда вы делаете коммит, т. е. сохраняете состояние своего проекта в Git, система запоминает, как выглядит каждый файл в этот момент, и сохраняет ссылку на этот снимок. Следует, что Git эффективен в хранении бэкапов, поэтому известно мало случаев, когда кто-то терял данные при его использовании.

### **5. Как обеспечивается целостность хранимых данных в Git?**

В Git для всего вычисляется хеш-сумма, и только потом происходит сохранение. В дальнейшем обращение к сохранённым объектам происходит



по этой хеш-сумме. Это значит, что невозможно изменить содержимое файла или директории так, чтобы Git не узнал об этом. Данная функциональность встроена в Git на низком уровне и является неотъемлемой частью его основы. В итоге информация не теряется во время её передачи и файл не повредится без ведома Git.

## **6. В каких состояниях могут находиться файлы в Git? Как связаны эти состояния?**

Зафиксированный файл — файл уже сохранён в вашей локальной базе.

Измененный файл — файл, который поменялся, но ещё не был зафиксирован.

Подготовленный файл — это изменённый файл, отмеченный для включения в следующий коммит.

## **7. Что такое профиль пользователя в GitHub?**

Профиль — ваша публичная страница на GitHub, как и в социальных сетях. Когда мы ищем работу в качестве программиста, работодатели могут посмотреть наш профиль GitHub и принять его во внимание, когда будут решать, брать нас на работу или нет.

## **8. Какие бывают репозитории в GitHub?**

Репозиторий Git бывает локальный и удалённый.

Локальный репозиторий — это подкаталог `.git`, создаётся (в пустом виде) командой `git init` и (в непустом виде с немедленным копированием содержимого родительского удалённого репозитория и простановкой ссылки

на родителя) командой `git clone`. Практически все обычные операции с системой контроля версий, такие, как коммит и слияние, производятся только с локальным репозиторием.

Удалённый доступ к репозиториям Git обеспечивается `git-daemon`, SSH-или HTTP-сервером. TCP-сервис `git-daemon` входит в дистрибутив Git и является наряду с SSH наиболее распространённым и надёжным методом доступа. Удалённый репозиторий можно только синхронизировать с локальным как «вверх» (`push`), так и «вниз» (`pull`).

### **9. Укажите основные этапы модели работы с GitHub.**

- 1) Регистрация.
- 2) Создание репозитория.
- 3) Клонирование репозитория

### **10. Как осуществляется первоначальная настройка Git после установки?**

- 1) Убедимся, что Git установлен используя команду: `git version`;
- 2) Перейдём в папку с локальным репозиторием, используя команду: `cd /с;`
- 3) Свяжем локальный репозиторий и удалённый командами:  
`git config --global user.name <YOUR_NAME>` `git config --global user.email <EMAIL>`

### **11. Опишите этапы создания репозитория в GitHub.**

1) В правом верхнем углу, рядом с аватаром есть кнопка с плюсиком, нажимая которую переходим к созданию нового репозитория.

2) В результате будет выполнен переход на страницу создания репозитория. Наиболее важными на ней являются следующие поля:

- Имя репозитория. Оно может быть любое, необязательно уникальное во всем github, потому что привязано к вашему аккаунту, но уникальное в рамках тех репозиториях, которые вы создавали.

- Описание (Description). Можно оставить пустым.

- Public/private. Выбираем открытый (Public), НЕ ставим галочку “Initialize this repository with a README” (в README потом будет лежать какая-то основная информация, что же такое ваш проект и как с ним работать).

- .gitignore и LICENSE можно сейчас не выбирать. После заполнения этих полей нажимаем кнопку Create repository.

## **12. Какие типы лицензий поддерживаются GitHub при создании репозитория?**

Microsoft Reciprocal License, The Code Project Open License (CPOPL), The Common Development and Distribution License (CDDL), The Microsoft Public License (Ms-PL), The Mozilla Public License 1.1 (MPL 1.1), The Common Public License Version 1.0 (CPL), The Eclipse Public License 1.0, The MIT License, The BSD License, The Apache License, Version 2.0, The Creative Commons Attribution-ShareAlike 2.5 License, The zlib/libpng License, A Public Domain dedication, The Creative Commons Attribution 3.0 Unported License, The Creative Commons Attribution-Share Alike 3.0 Unported License, The Creative Commons Attribution-NoDerivatives 3.0 Unported, The GNU Lesser General Public License (LGPLv3), The GNU General Public License (GPLv3).

**13. Как осуществляется клонирование репозитория GitHub?  
Зачем нужно клонировать репозиторий?**

Используя команду: `git status`.

**14. Как проверить состояние локального репозитория Git?**

Файлы обновятся на удалённом репозитории.

**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` ?**

Файлы обновятся на удалённом 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` .**

Примечание: описание необходимо начать с команды `git clone` .

1) Клонировем репозиторий на каждый из компьютеров, используя команду `git clone` и ссылку.

2) Для синхронизации изменений используем команду `git pull`.

**17. GitHub является не единственным сервисом, работающим с Git. Какие сервисы еще Вам известны? Приведите сравнительный анализ одного из таких сервисов с GitHub.**

GitLab — альтернатива GitHub номер один. GitLab предоставляет не только веб-сервис для совместной работы, но и программное обеспечение с открытым исходным кодом.

SourceForge — ещё одна крупная альтернатива GitHub, сконцентрировавшаяся на Open Source. Многие дистрибутивы и приложения Linux обитают на SourceForge

Launchpad — платформа для совместной работы над программным обеспечением от Canonical, компании-разработчика Ubuntu. На ней размещены PPA-репозитории Ubuntu, откуда пользователи загружают приложения и обновления.

**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.**

1) GitHub Desktop это бесплатное приложение с открытым исходным кодом, разработанное GitHub. С его помощью можно взаимодействовать с GitHub, а также с другими платформами (включая GitLab).

2) Fork это весьма продвинутый GUI-клиент для macOS и Windows (с бесплатным пробным периодом). В фокусе этого инструмента скорость, дружелюбность к пользователю и эффективность. К особенностям Fork

можно отнести красивый вид, кнопки быстрого доступа, встроенную систему разрешения конфликтов слияния, менеджер репозитория, уведомления GitHub.

3) Sourcetree это бесплатный GUI Git для macOS и Windows. Его применение упрощает работу с контролем версий и позволяет сфокусироваться на действительно важных задачах.

4) martGit это Git-клиент для Mac, Linux и Windows. Имеет богатый функционал. В арсенале SmartGit вы найдете CLI для Git, графическое отображение слияний и истории коммитов, SSH-клиент, Git-Flow, программу для разрешения конфликтов слияния.

**Вывод:** в ходе лабораторной работы я исследовал базовые возможности системы контроля версий Git и веб-сервиса для хостинга IT проектов GitHub. Создал GitHub репозиторий, клонировал репозиторий на компьютер, написал небольшую программу и отправил изменения на удалённый репозиторий.