

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

Институт цифрового развития
Кафедра инфокоммуникаций

ОТЧЕТ
ПО РАБОТЕ №1.2
дисциплины «Основы кроссплатформенного программирования»

Выполнил:

Хачатрян Владимир Владимирович

1 курс, группа ИТС-б-о-22-1,

11.03.02 «Инфокоммуникационные
технологии и системы связи»,
направленность (профиль)

«Инфокоммуникационные системы и

сети», очная форма обучения

(подпись)

Руководитель практики:

Воронкин Р.А., канд. тех. наук, доцент,
доцент кафедры инфокоммуникаций

(подпись)

Отчет защищен с оценкой _____ Дата защиты _____

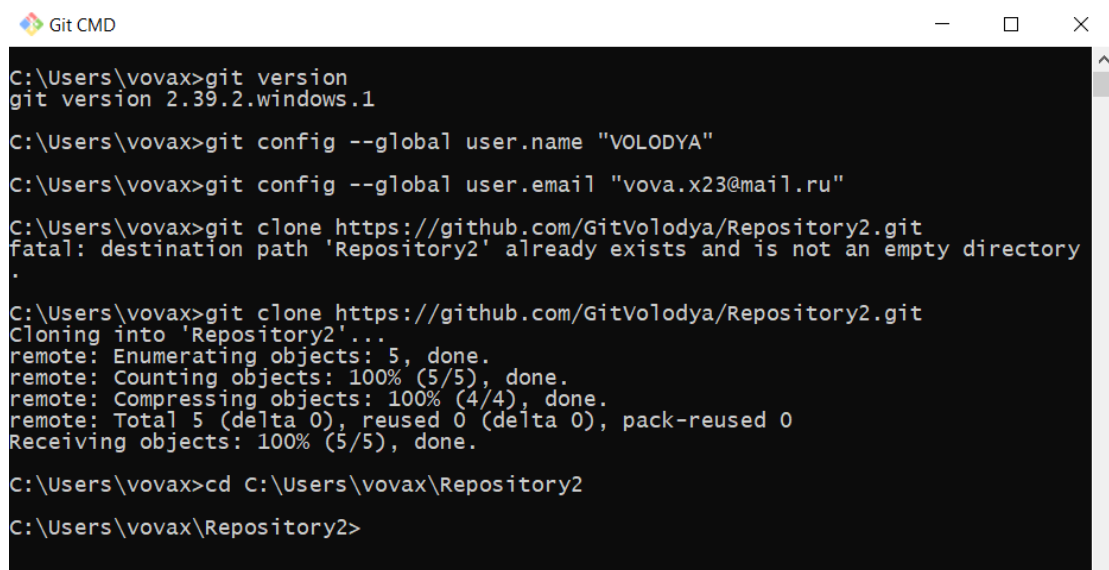
Ставрополь, 2023 г.

Тема: исследование возможностей Git для работы с локальными репозиториями.

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

Порядок выполнения работы:

Задание 1. Создал новый репозиторий и клонировал его на свой компьютер.



```
Git CMD
C:\Users\vovax>git version
git version 2.39.2.windows.1

C:\Users\vovax>git config --global user.name "VOLODYA"

C:\Users\vovax>git config --global user.email "vova.x23@mail.ru"

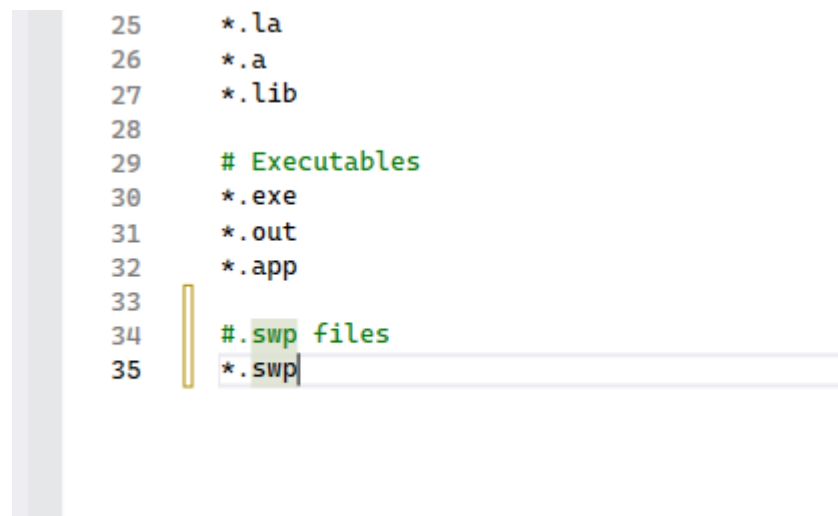
C:\Users\vovax>git clone https://github.com/GitVolodya/Repository2.git
fatal: destination path 'Repository2' already exists and is not an empty directory

C:\Users\vovax>git clone https://github.com/GitVolodya/Repository2.git
Cloning into 'Repository2'...
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
Receiving objects: 100% (5/5), done.

C:\Users\vovax>cd C:\Users\vovax\Repository2
C:\Users\vovax\Repository2>
```

Рисунок 1. Новый репозиторий

Задание 2. Добавил некоторое правило в файл gitignore, чтобы Git игнорировал файлы в формате .swp.



```
25 *.la
26 *.a
27 *.lib
28
29 # Executables
30 *.exe
31 *.out
32 *.app
33
34 #.swp files
35 *.swp
```

Рисунок 2. Работа с gitignore

Задание 3. Добавил информацию в файл README.md о дисциплине, группе и ФИО.

```
# Repository2
Osnovy crossplatformennogo programmirovaniya
ITS-b-o-22-1 Khachatryan Vladimir|
```

Рисунок 3. Работа с README

Задание 4. Написал программу в новом файле main.cpp, сделал 7 коммитов с 3-мя тегами.

```
C:\Users\vovax\Repository2>git log --graph --pretty=oneline --abbrev-commit
* 20022fff (HEAD -> main, tag: End) com7
* 064552f com6
* 8fe5ae5 com5
* 50a1897 (tag: Middle) com4
* 55d7272 com3
* ffb13d1 com2
* 4c7fd19 (tag: Nachalo) com1
* c2a271f (origin/main, origin/HEAD) Initial commit
C:\Users\vovax\Repository2>
```

Рисунок 4. История хранилища

С помощью команды «git log --graph --pretty=oneline --abbrev-commit» можно проанализировать все свои коммиты и теги.

Задание 5. Посмотрел содержимое коммитов командой git show <ref>, где <ref>:

1) HEAD : последний коммит;

```
C:\Users\vovax\Repository2>git show HEAD
commit 20022ffca3242c5dd1028ac926ea25524c4f5732 (HEAD -> main, tag: End)
Author: VOLODYA <vova.x23@mail.ru>
Date:   Wed Mar 15 12:27:17 2023 +0300

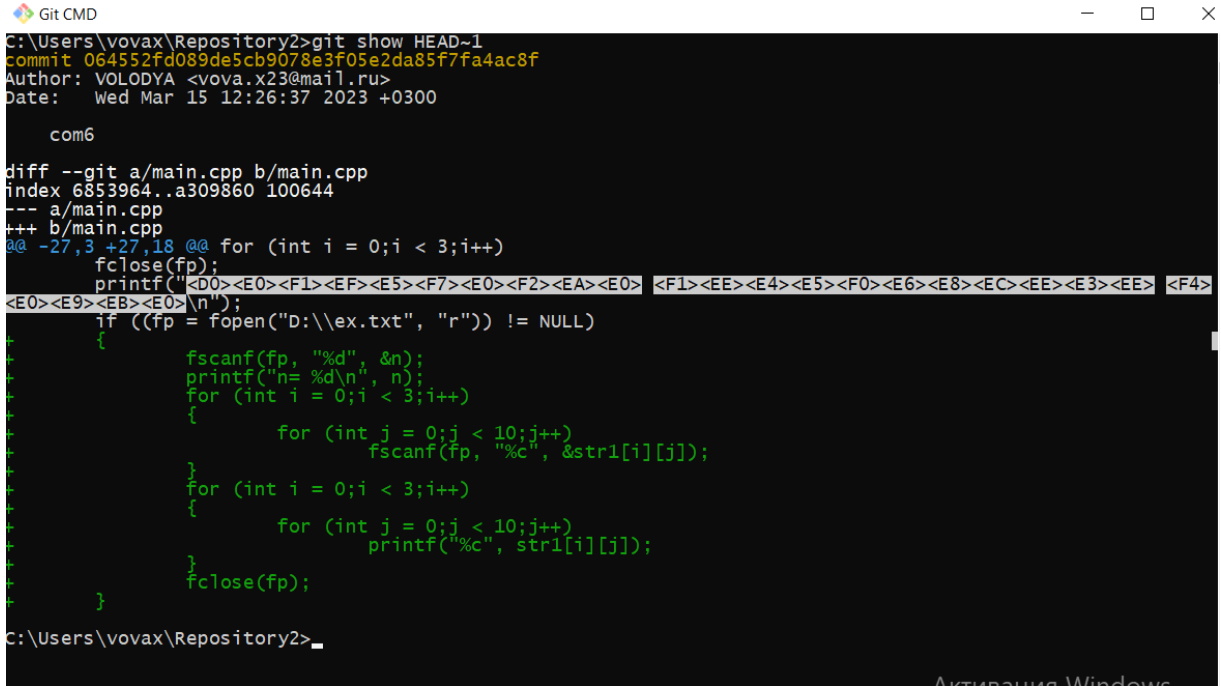
    com7

diff --git a/main.cpp b/main.cpp
index a309860..e063acf 100644
--- a/main.cpp
+++ b/main.cpp
@@ -42,3 +42,7 @@ for (int i = 0; i < 3; i++)
    }
    fclose(fp);
+
+    else
+        printf("\n <CD><E5><EB><FC><E7><FF> <EE><F2><EA><F0><FB><F2><FC> <F4><E0><E9><EB>
+<E4><EB><FF> <F7><F2><E5><ED><E8><FF> !");
+        getch();
+}

C:\Users\vovax\Repository2>
```

Рисунок 5. Последний коммит

2) HEAD~1 : предпоследний коммит (и т. д.);



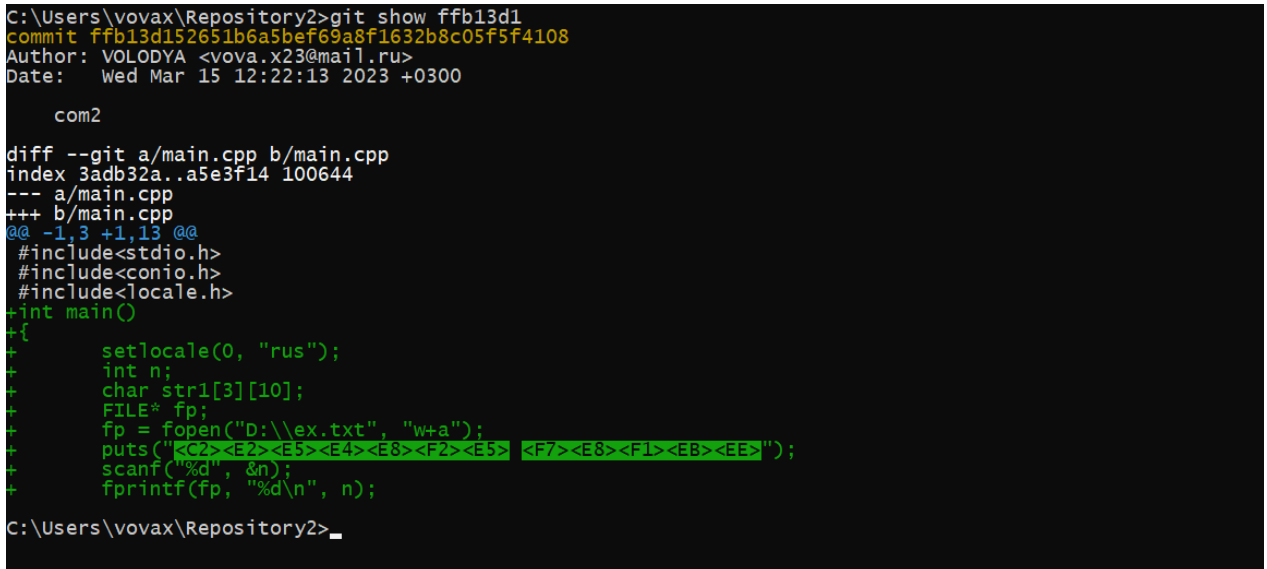
```
Git CMD
C:\Users\vovax\Repository2>git show HEAD~1
commit 064552fd089de5cb9078e3f05e2da85f7fa4ac8f
Author: VOLODYA <vova.x23@mail.ru>
Date:   Wed Mar 15 12:26:37 2023 +0300

    com6

diff --git a/main.cpp b/main.cpp
index 6853964..a309860 100644
--- a/main.cpp
+++ b/main.cpp
@@ -27,3 +27,18 @@
     fclose(fp);
     printf("\n");
     if ((fp = fopen("D:\\ex.txt", "r")) != NULL)
     {
         fscanf(fp, "%d", &n);
         printf("n= %d\n", n);
         for (int i = 0; i < 3; i++)
         {
             for (int j = 0; j < 10; j++)
                 fscanf(fp, "%c", &str1[i][j]);
         }
         for (int i = 0; i < 3; i++)
         {
             for (int j = 0; j < 10; j++)
                 printf("%c", str1[i][j]);
             fclose(fp);
         }
     }
C:\Users\vovax\Repository2>
```

Рисунок 6. Предпоследний коммит.

3) b34a0e : коммит с указанным хэшем.



```
C:\Users\vovax\Repository2>git show ffb13d1
commit ffb13d152651b6a5bef69a8f1632b8c05f5f4108
Author: VOLODYA <vova.x23@mail.ru>
Date:   Wed Mar 15 12:22:13 2023 +0300

    com2

diff --git a/main.cpp b/main.cpp
index 3adb32a..a5e3f14 100644
--- a/main.cpp
+++ b/main.cpp
@@ -1,3 +1,13 @@
#include<stdio.h>
#include<conio.h>
#include<locale.h>
+int main()
+{
+    setlocale(0, "rus");
+    int n;
+    char str1[3][10];
+    FILE* fp;
+    fp = fopen("D:\\ex.txt", "w+a");
+    puts("n= ");
+    scanf("%d", &n);
+    fprintf(fp, "%d\n", n);
C:\Users\vovax\Repository2>
```

Рисунок 7. Коммит с указанным хэшем.

Задание 6. Откат к заданной версии.

1.1. Удалил весь код из файла main.cpp и сохранил его.

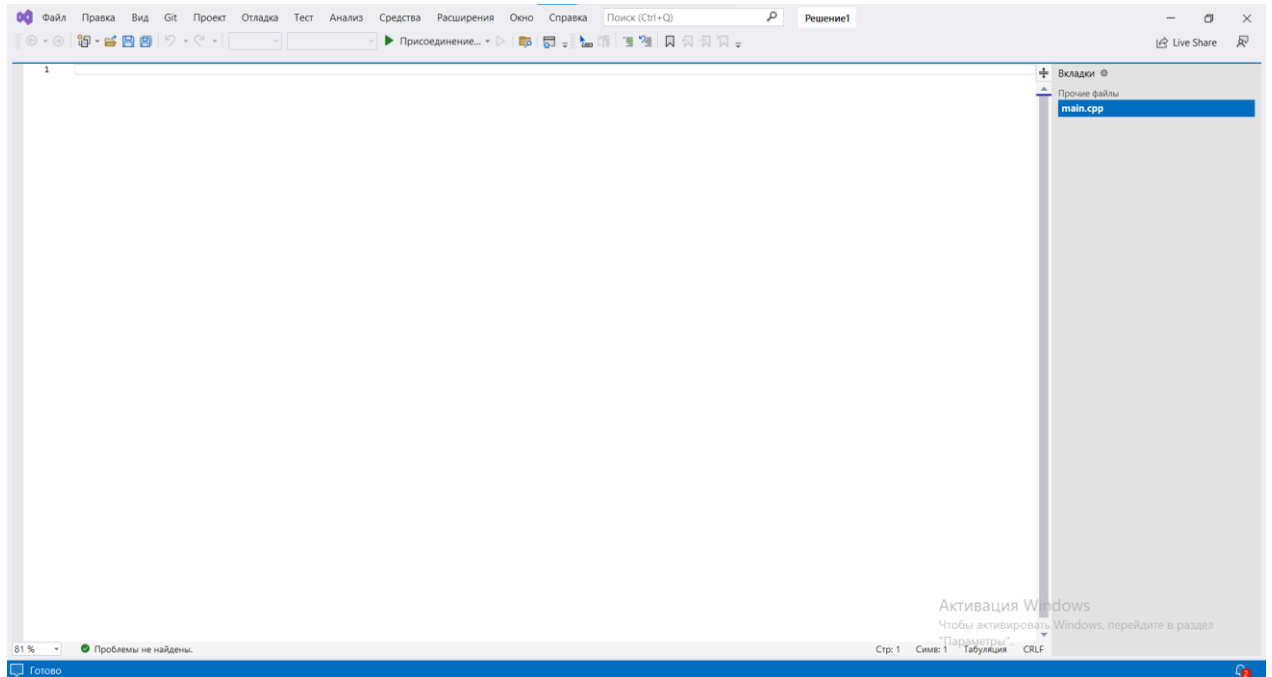


Рисунок 8. Удаление программ

Удалил это изменение с помощью команды «git checkout -- main.cpp».

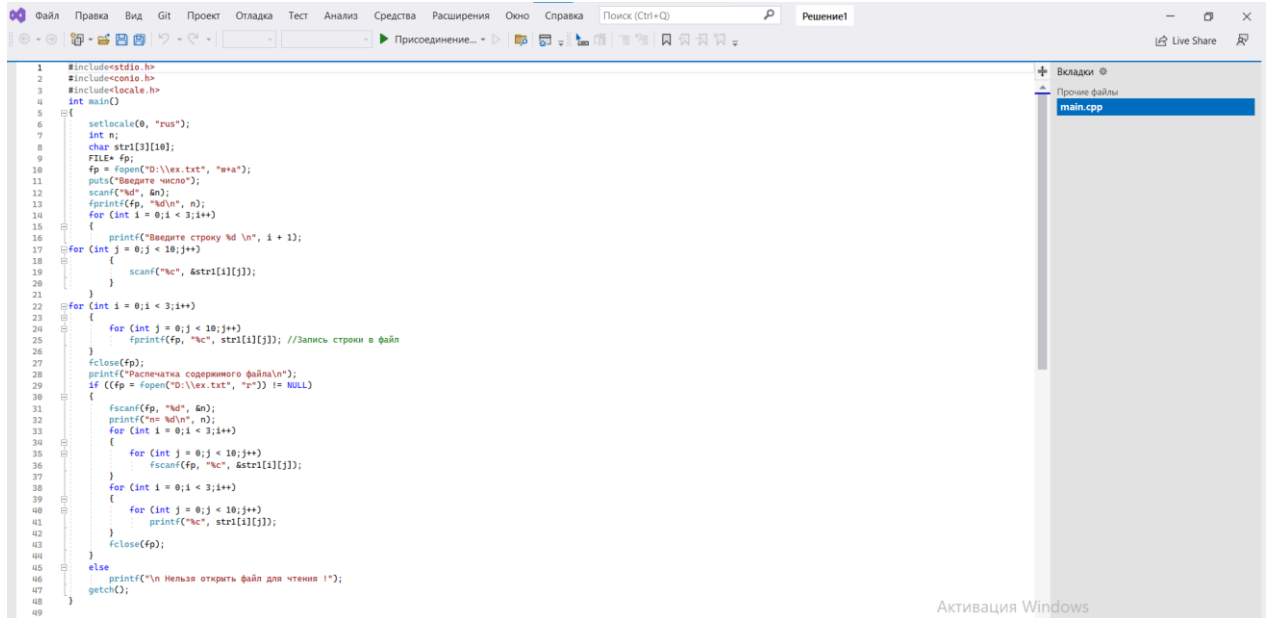


Рисунок 9. Восстановление программы.

1.2. Повторил пункт 1.1. и сделал коммит.

```
C:\Users\vovax\Repository2>git add .  
C:\Users\vovax\Repository2>git commit -m "Delete"  
[main 2846e1d] Delete  
1 file changed, 1 insertion(+), 48 deletions(-)  
C:\Users\vovax\Repository2>
```

Рисунок 10. Коммит

1.3. Откатил состояние хранилища к предыдущей версии командой:
«git reset --hard HEAD~1».

```
C:\Users\vovax\Repository2>git reset --hard HEAD~1  
HEAD is now at 20022ff com7  
C:\Users\vovax\Repository2>_
```

Рисунок 10. Возвращение к предпоследней версии коммита

Ссылка на репозиторий: <https://github.com/GitVolodya/Repository2.git>

Ответы на контрольные вопросы:

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

Историю коммитов можно выполнить с помощью команды git log.

Дополнительные опции для просмотра истории: %H,

%h, %T, %t, %P, %p тд.

-p, --stat, --shortstat, --name-only, --name-status и тд.

2) *Как ограничить вывод при просмотре истории коммитов?*

Ограничить вывод при просмотре истории коммитов можно с помощью команды «git log -n», где n – число последних коммитов.

3) *Как внести изменения в уже сделанный коммит?*

Если вы хотите переделать коммит — внесите необходимые изменения, добавьте их в индекс и сделайте коммит ещё раз, указав параметр «--amend» : «git commit –amend».

4) *Как отменить индексацию файла в Git?*

Отменить индексацию файла можно с помощью команды: «git reset HEAD <file>».

5) *Как отменить изменения в файле?*

Отменить изменения в файле можно с помощью команды: «git checkout - <file>»

6) *Что такое удаленный репозиторий Git?*

Удалённые репозитории представляют собой версии вашего проекта, сохранённые в интернете или ещё где-то в сети.

7) *Как выполнить просмотр удаленных репозиториях данного локального репозитория?*

Выполнить просмотр удаленных репозиториях данного локального репозитория можно с помощью команды: *git remote*.

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 <shortname> <url>».

9) *Как выполнить отправку/получение изменений с удаленного репозитория?*

Для получения данных из удалённых проектов, следует выполнить: «git fetch [remote-name]».

Для отправки изменений в удаленный репозиторий используется команда: «git push <remote-name> <branch-name>».

10) *Как выполнить просмотр удаленного репозитория?*

Если хотите получить побольше информации об одном из удалённых репозиториях, вы можете использовать команду: «git remote show <remote>».

11) *Каково назначение тэгов Git?*

Git имеет возможность пометить определённые моменты в истории как важные. Для таких случаев были придуманы тэги.

12) *Как осуществляется работа с тэгами Git?*

Просмотреть список имеющихся тегов в Git можно очень просто. Достаточно набрать команду «git tag».

Создание аннотированного тега в Git выполняется легко. Самый простой способ — это указать *-a* при выполнении команды «tag».

С помощью команды «git show» вы можете посмотреть данные тега вместе с коммитом.

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 <tagname>».

Для удаления тега в локальном репозитории достаточно выполнить команду «git tag -d <tagname>» .

Если вы хотите получить версии файлов, на которые указывает тег, то вы можете сделать «git checkout <tagname>» для тега.

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

«Git prune» – это команда, которая удаляет все файлы, недоступные из текущей ветки. Команда «prune» полезна, когда в вашем рабочем каталоге много файлов, которые вы не хотите хранить. «git fetch --prune» делает то же

самое: удалет ссылки на ветки, которые не существуют на удаленном компьютере.

Опция «`-prune`» в команде «`git push`» удалит ветку из удаленного репозитория, если в локальном репозитории не существует ветки с таким именем.

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