

Звіт

З лабораторної роботи №1

Студента групи МІТ-31

Добровольського Арсенія Михайловича

### Тема роботи: Основи Git

**Мета роботи:** ознайомитися з призначенням та функціональними можливостями системи контролю версій Git та навчитися виконувати базові операції в ній.

### Хід виконання роботи

Виклик команди git:

```
cmd Command Prompt
Microsoft Windows [Version 10.0.19045.3324]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Fijitsu E546>git
usage: git [-v | --version] [-h | --help] [-C <path>] [-c <name>=<value>]
          [--exec-path[=<path>]] [--html-path] [--man-path] [--info-path]
          [-p | --paginate | -P | --no-pager] [--no-replace-objects] [--bare]
          [--git-dir=<path>] [--work-tree=<path>] [--namespace=<name>]
          [--config-env=<name>=<envvar>] <command> [<args>]

These are common Git commands used in various situations:


start a working area (see also: git help tutorial)
  clone      Clone a repository into a new directory
  init       Create an empty Git repository or reinitialize an existing one

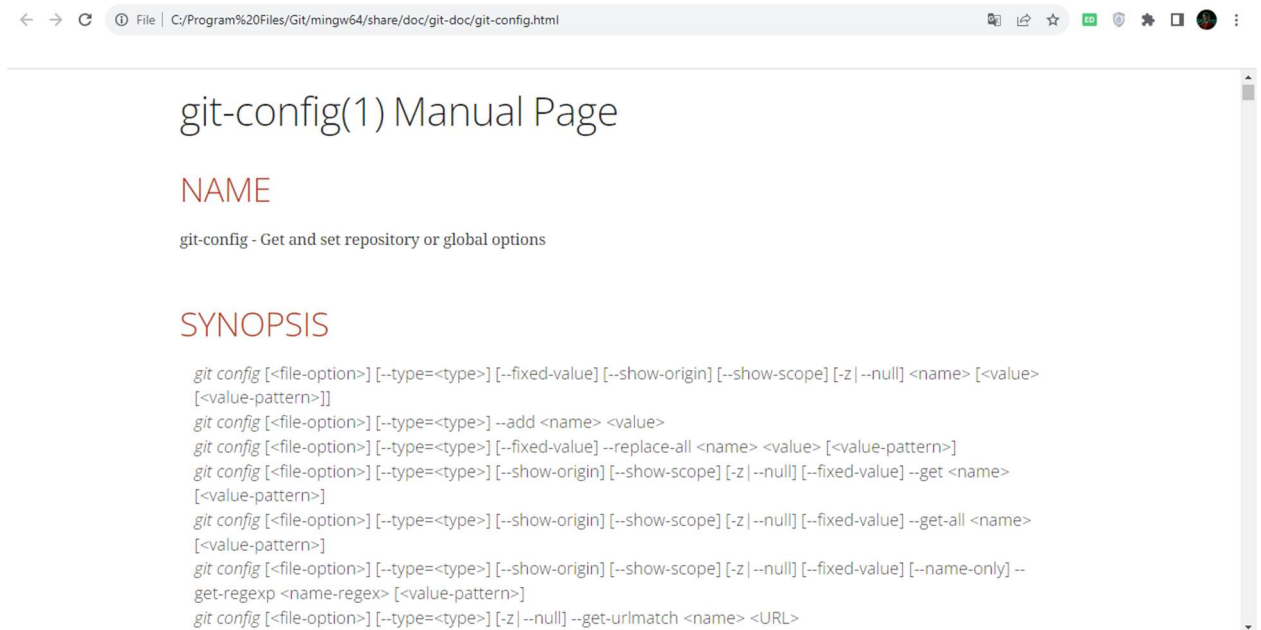

work on the current change (see also: git help everyday)
  add        Add file contents to the index
  mv         Move or rename a file, a directory, or a symlink
  restore    Restore working tree files
  rm         Remove files from the working tree and from the index


examine the history and state (see also: git help revisions)
  bisect    Use binary search to find the commit that introduced a bug
  diff      Show changes between commits, commit and working tree, etc
  grep      Print lines matching a pattern
  log       Show commit logs
  show      Show various types of objects
  status    Show the working tree status


grow, mark and tweak your common history
  branch    List, create, or delete branches
  commit    Record changes to the repository
  merge     Join two or more development histories together
  rebase    Reapply commits on top of another base tip
  reset     Reset current HEAD to the specified state
  switch    Switch branches
  tag       Create, list, delete or verify a tag object signed with GPG


collaborate (see also: git help workflows)
  fetch     Download objects and refs from another repository
  pull      Fetch from and integrate with another repository or a local branch
  push      Update remote refs along with associated objects
```

Виклик довідки для команди `config` (команди `git config --help` та `git help config`):



В результаті виконання обох команд у вікні браузера відображається довідкова сторінка по команді `config`, яка містить детальний опис цієї команди, можливі параметри, приклади її використання тощо.

Налаштування імені користувача та адреси електронної пошти у глобальному файлі конфігурації:

```
C:\Users\Fijitsu E546>git config --global user.name "Arsenii Dobrovolskyi"
C:\Users\Fijitsu E546>git config --global user.email cleverboyars@gmail.com
```

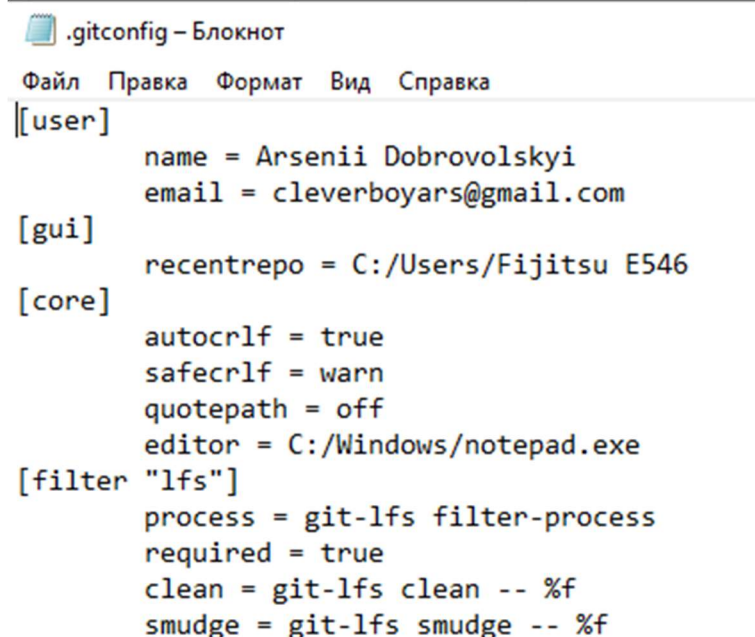
Налаштування текстового редактора за замовчуванням:

```
C:\Users\Fijitsu E546>cd ../../Program Files/Git
C:\Program Files\Git>git config --global core.editor "C:/Windows/notepad.exe"
```

Перевірка налаштувань Git:

```
C:\Program Files\Git>git config --list
diff.astextplain.textconv=astextplain
filter.lfs.clean=git-lfs clean -- %f
filter.lfs.smudge=git-lfs smudge -- %f
filter.lfs.process=git-lfs filter-process
filter.lfs.required=true
http.sslbackend=openssl
http.sslcainfo=C:/Program Files/Git/mingw64/etc/ssl/certs/ca-bundle.crt
core.autocrlf=true
core.fscache=true
core.symlinks=false
core.usebuiltinfsmonitor=true
core.fsmonitor=true
pull.rebase=false
credential.helper=manager
credential.https://dev.azure.com.usehttppath=true
init.defaultbranch=main
user.name=Arsenii Dobrovolskyi
user.email=cleverboyars@gmail.com
gui.recentrepo=C:/Users/Fijitsu E546
core.autocrlf=true
core.safecrlf=warn
core.quoteopath=off
core.editor=C:/Windows/notepad.exe
filter.lfs.process=git-lfs filter-process
filter.lfs.required=true
filter.lfs.clean=git-lfs clean -- %f
filter.lfs.smudge=git-lfs smudge -- %f
```

Основний конфігураційний файл має вигляд:



The screenshot shows a Notepad window titled ".gitconfig - Блокнот". The menu bar includes "Файл", "Правка", "Формат", "Вид", and "Справка". The content of the file is as follows:

```
[user]
    name = Arsenii Dobrovolskyi
    email = cleverboyars@gmail.com

[gui]
    recentrepo = C:/Users/Fijitsu E546

[core]
    autocrlf = true
    safecrlf = warn
    quoteopath = off
    editor = C:/Windows/notepad.exe

[filter "lfs"]
    process = git-lfs filter-process
    required = true
    clean = git-lfs clean -- %f
    smudge = git-lfs smudge -- %f
```

Дані, записані у цьому файлі, свідчать про те, що всі налаштування виконано правильно.



Створення комітів проєкту testApp та перегляд історії комітів наведено на скріншоті нижче:

```
D:\KNU_FIT\3rd_grade\Programming technologies\Lab1\testApp>git commit
[main (root-commit) 33d72dc] My first commit
10 files changed, 114 insertions(+)
create mode 100644 .gitignore
create mode 100644 .idea/.gitignore
create mode 100644 .idea/compiler.xml
create mode 100644 .idea/encodings.xml
create mode 100644 .idea/jarRepositories.xml
create mode 100644 .idea/misc.xml
create mode 100644 .idea/vcs.xml
create mode 100644 pom.xml
create mode 100644 readme.txt
create mode 100644 src/main/java/testapp/TestApp.java

D:\KNU_FIT\3rd_grade\Programming technologies\Lab1\testApp>git commit -a -m "Calculations added"
[main 2659cdd] Calculations added
1 file changed, 3 insertions(+)

D:\KNU_FIT\3rd_grade\Programming technologies\Lab1\testApp>git add --all

D:\KNU_FIT\3rd_grade\Programming technologies\Lab1\testApp>git commit -a -m "Scanner feature added"
[main 5717fa5] Scanner feature added
1 file changed, 7 insertions(+), 2 deletions(-)

D:\KNU_FIT\3rd_grade\Programming technologies\Lab1\testApp>git log
commit 5717fa57c99e16a6270c2af111b6135bd126f212 (HEAD -> main)
Author: Arsenii Dobrovolskyi <cleverboyars@gmail.com>
Date: Sun Sep 17 17:36:14 2023 +0300

    Scanner feature added

commit 2659cddf28a0dff203fd6fca63856e4f46330c78
Author: Arsenii Dobrovolskyi <cleverboyars@gmail.com>
Date: Sun Sep 17 16:47:36 2023 +0300

    Calculations added

commit 33d72dc8b06bfdb79717b82dc86f1a41d694d3fb
Author: Arsenii Dobrovolskyi <cleverboyars@gmail.com>
Date: Sun Sep 17 16:38:52 2023 +0300

    My first commit
```

До основних параметрів команди git log належать:

- -p (показує різницю, привнесену кожним комітом)
- --stat (коротка статистика змін, що включає кількість внесених змін, вставок та вилучень файлів для кожного коміту)
- --oneline (виводить кожен коміт в одному рядку, представляючи ідентифікатор коміту (SHA-1 хеш) і його короткий опис)
- --graph (виводить графічне представлення гілок і злиттів у історії комітів)
- --author="Ім'я автора" (фільтрує коміти за автором)
- -n (n – довільне число) – виводить вказану кількість комітів
- --grep="текст пошуку" (фільтрує коміти за текстом повідомлення коміту)

## Завдання для самостійної роботи

Для виконання команд, що розглянуті далі, за основу взято репозиторій, створений в рамках вивчення минулорічної дисципліни.

### Ознайомлення з командами git

#### a) git clone

Ця команда використовується для створення копії віддаленого репозиторію Git на локальному комп'ютері.

```
MINGW64:/d/KNU_FIT/3rd_grade/Programming technologies/Lab1
Ars@DESKTOP-3S04H0P MINGW64 /d/KNU_FIT/3rd_grade/Programming technologies/Lab1
$ git clone https://github.com/arsgo00/InfSec_MIT21.git
Cloning into 'InfSec_MIT21'...
remote: Enumerating objects: 760, done.
remote: Counting objects: 100% (760/760), done.
remote: Compressing objects: 100% (366/366), done.
remote: Total 760 (delta 345), reused 624 (delta 217), pack-reused 0
Receiving objects: 86% (654/760)
Receiving objects: 100% (760/760), 1.63 MiB | 3.11 MiB/s, done.
Resolving deltas: 100% (345/345), done.
```

#### b) git status

Надає інформацію про поточний стан локального репозиторію.

Наприклад, вона відображає файли які було змінено, але не проіндексовано, файли, які вже містяться в індексі і очікують на потрапляння в коміт тощо.

Для того, щоб побачити роботу команди наяву, зроблю зміни в одному з файлів свого репозиторію. Таким чином команда git status виведе такий результат:

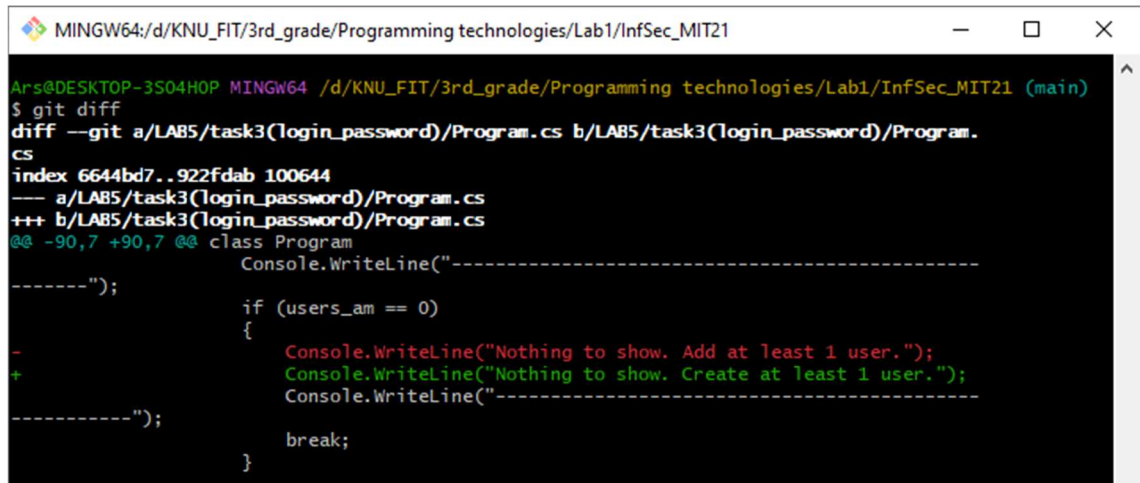
```
MINGW64:/d/KNU_FIT/3rd_grade/Programming technologies/Lab1/InfSec_MIT21
Ars@DESKTOP-3S04H0P MINGW64 /d/KNU_FIT/3rd_grade/Programming technologies/Lab1/InfSec_MIT21 (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   LAB5/task3(login_password)/Program.cs

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

### c) git diff

Виявляє та відображає різницю між двома будь-якими станами репозиторію. Зокрема, за допомогою цієї команди можна побачити різницю між робочою директорією та індексом, між індексом та останнім комітом (`git diff --staged`), між двома комітами або навіть гілками. Фактично `git diff` деталізує інформацію, яку відображає команда `git status`, тобто показує не лише файли, які було змінено, а й конкретні зміни:

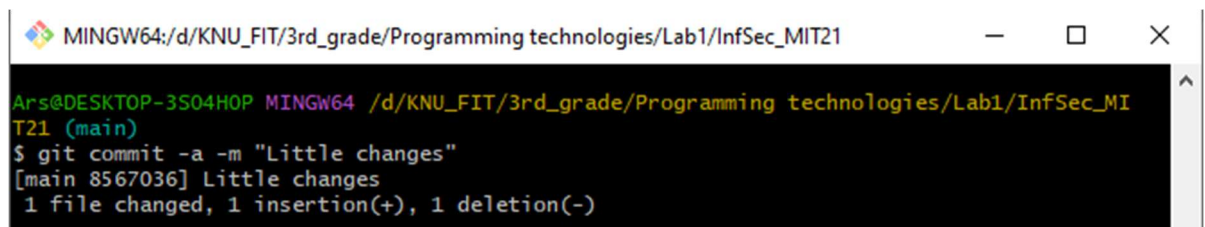


```
MINGW64:/d/KNU_FIT/3rd_grade/Programming technologies/Lab1/InfSec_MIT21
Ars@DESKTOP-3S04H0P MINGW64 /d/KNU_FIT/3rd_grade/Programming technologies/Lab1/InfSec_MIT21 (main)
$ git diff
diff --git a/LAB5/task3(login_password)/Program.cs b/LAB5/task3(login_password)/Program.cs
index 6644bd7..922fdab 100644
--- a/LAB5/task3(login_password)/Program.cs
+++ b/LAB5/task3(login_password)/Program.cs
@@ -90,7 +90,7 @@ class Program
     Console.WriteLine("-----");
-----");
        if (users_am == 0)
        {
-           Console.WriteLine("Nothing to show. Add at least 1 user.");
+           Console.WriteLine("Nothing to show. Create at least 1 user.");
           Console.WriteLine("-----");
        }
        break;
    }
```

В той час як команда `git status` відобразила лише те, що був змінений файл `Program.cs`, `git diff` показує, яких саме змін зазнав цей файл.

### d) git commit

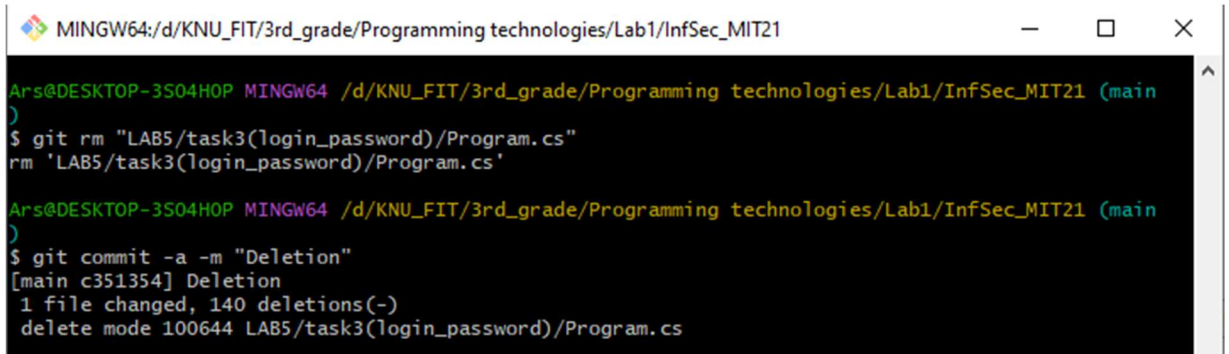
Створює новий коміт, тобто зберігає зміни, зроблені у локальному репозиторії. Перед цим слід проіндексувати файли, виконавши команду `git add`. або можна просто додати параметр `-a` до команди `git commit`. Також стане в пригоді параметр `-m`, що дозволяє додати коментар до коміту без відкриття текстового редактора.



```
MINGW64:/d/KNU_FIT/3rd_grade/Programming technologies/Lab1/InfSec_MIT21
Ars@DESKTOP-3S04H0P MINGW64 /d/KNU_FIT/3rd_grade/Programming technologies/Lab1/InfSec_MIT21 (main)
$ git commit -a -m "Little changes"
[main 8567036] Little changes
1 file changed, 1 insertion(+), 1 deletion(-)
```

e) git rm

Видаляє файли або папки з індексу та робочої директорії. Для демонстрації роботи цієї команди видалю файл, в який було внесено зміни у попередніх підпунктах та виконаю команду git commit для збереження змін:



```
MINGW64:/d/KNU_FIT/3rd_grade/Programming technologies/Lab1/InfSec_MIT21
Ars@DESKTOP-3S04H0P MINGW64 /d/KNU_FIT/3rd_grade/Programming technologies/Lab1/InfSec_MIT21 (main)
$ git rm "LAB5/task3(login_password)/Program.cs"
rm 'LAB5/task3(login_password)/Program.cs'

Ars@DESKTOP-3S04H0P MINGW64 /d/KNU_FIT/3rd_grade/Programming technologies/Lab1/InfSec_MIT21 (main)
$ git commit -a -m "Deletion"
[main c351354] Deletion
1 file changed, 140 deletions(-)
delete mode 100644 LAB5/task3(login_password)/Program.cs
```

f) git mv

Переміщає або перейменовує файли в репозиторії. Виконання цієї команди також включає видалення вихідного файлу (git rm) та додавання в індекс нового (git add):



```
MINGW64:/d/KNU_FIT/3rd_grade/Programming technologies/Lab1/InfSec_MIT21
Ars@DESKTOP-3S04H0P MINGW64 /d/KNU_FIT/3rd_grade/Programming technologies/Lab1/InfSec_MIT21 (main)
$ git mv LAB5/LAB5.sln LAB6

Ars@DESKTOP-3S04H0P MINGW64 /d/KNU_FIT/3rd_grade/Programming technologies/Lab1/InfSec_MIT21 (main)
$ git status
On branch main
Your branch is ahead of 'origin/main' by 2 commits.
(use "git push" to publish your local commits)

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        renamed:   LAB5/LAB5.sln -> LAB6/LAB5.sln

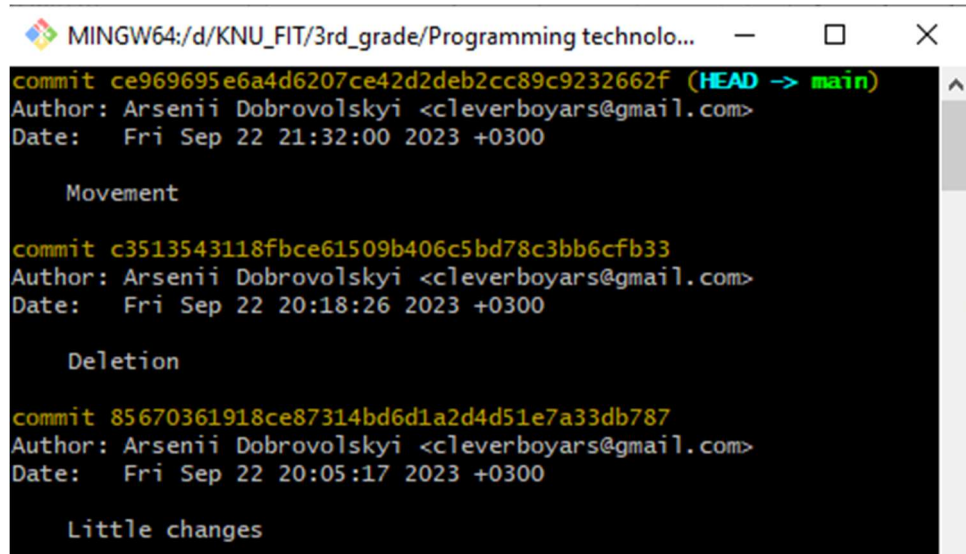
Ars@DESKTOP-3S04H0P MINGW64 /d/KNU_FIT/3rd_grade/Programming technologies/Lab1/InfSec_MIT21 (main)
$ git commit -a -m "Movement"
[main ce96969] Movement
1 file changed, 0 insertions(+), 0 deletions(-)
rename {LAB5 => LAB6}/LAB5.sln (100%)
```

Слід зазначити, що на відміну від багатьох систем контролю версій, Git не відстежує переміщення файлів явно, тому він вважає, що відбулося перейменування файлу (про це свідчить результат виконання команди git status).



### g) git log

Відображає історію комітів, починаючи від найновішого до найстарішого. Ця команда також виводить корисну інформацію про кожен коміт, таку як хеш-код, автор, дата, час створення і коментар:



```
MINGW64:/d/KNU_FIT/3rd_grade/Programming technolo...
commit ce969695e6a4d6207ce42d2deb2cc89c9232662f (HEAD -> main)
Author: Arsenii Dobrovolskyi <cleverboyars@gmail.com>
Date:   Fri Sep 22 21:32:00 2023 +0300

    Movement

commit c3513543118fbce61509b406c5bd78c3bb6cfb33
Author: Arsenii Dobrovolskyi <cleverboyars@gmail.com>
Date:   Fri Sep 22 20:18:26 2023 +0300


    Deletion

commit 85670361918ce87314bd6d1a2d4d51e7a33db787
Author: Arsenii Dobrovolskyi <cleverboyars@gmail.com>
Date:   Fri Sep 22 20:05:17 2023 +0300

    Little changes
```

### h) git cat-file

Дозволяє переглядати вміст та різні корисні властивості об'єктів у репозиторії. До об'єктів відносяться коміти, дерева, теги та блоки. Якщо до команди додати параметр `-t`, на екран виведеться тип об'єкта, а якщо `-p`, то його вміст:



```
MINGW64:/d/KNU_FIT/3rd_grade/Programming technologies/Lab1/InfSec_MIT21
Ars@DESKTOP-3S04H0P MINGW64 /d/KNU_FIT/3rd_grade/Programming technologies/Lab1/InfSec_MIT21 (main)
$ git cat-file -t ce969695e6a4d6207ce42d2deb2cc89c9232662f
commit

Ars@DESKTOP-3S04H0P MINGW64 /d/KNU_FIT/3rd_grade/Programming technologies/Lab1/InfSec_MIT21 (main)
$ git cat-file -p ce969695e6a4d6207ce42d2deb2cc89c9232662f
tree 4e9ed22d845f07e5278a162e14482560750afdf0
parent c3513543118fbce61509b406c5bd78c3bb6cfb33
author Arsenii Dobrovolskyi <cleverboyars@gmail.com> 1695407520 +0300
committer Arsenii Dobrovolskyi <cleverboyars@gmail.com> 1695407520 +0300

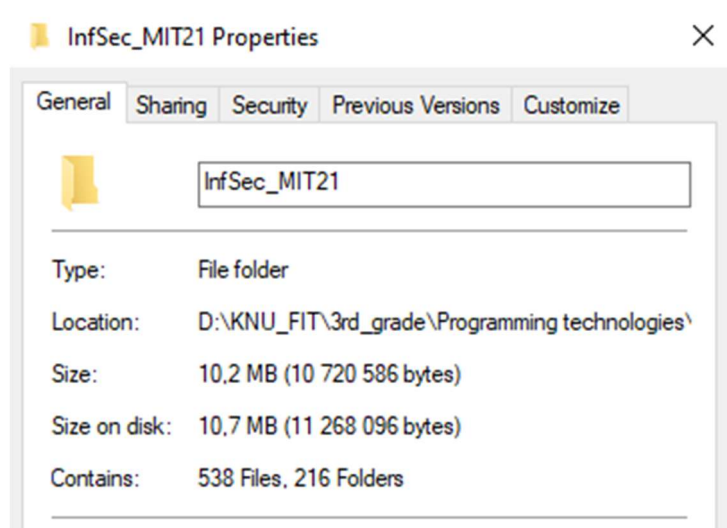
Movement
```

### Додаткове завдання

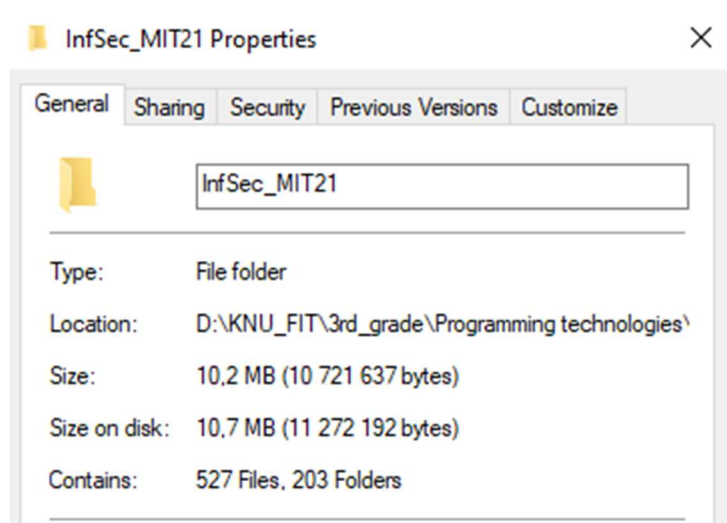
В Git спосіб зберігання даних не залежить від розміру окремого файлу в репозиторії. Він обробляє файли як послідовності байтів і стежить за змінами вмісту файлів, а не їхнім розміром. Розмір самого файлу впливає лише на обсяг даних, які Git повинен зберігати, але не на спосіб їх зберігання.

Команда `git gc` використовується для оптимізації роботи Git-репозиторію шляхом видалення непотрібних та застарілих файлів зі сховища об'єктів і упакуванні решти файлів, тим самим зменшуючи розмір репозиторію.

Початковий розмір репозиторію:



Розмір репозиторію після виконання команди `git gc`:



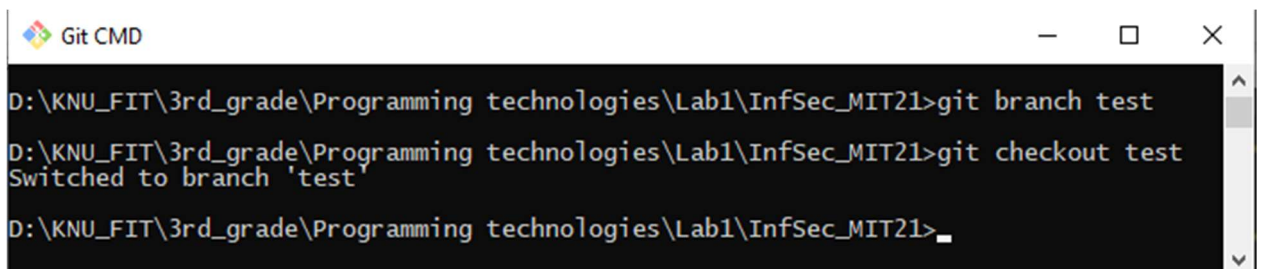
## Частина №2

**Тема роботи:** Робота з різними гілками програми в Git. Конфлікти в Git.

**Мета роботи:** навчитися працювати з різними гілками програми в Git, розглянути причини виникнення конфліктів зливання у Git та засоби боротьби з ними.

### Завдання для самостійної роботи

Спершу створюю нову гілку test у репозиторії з першої частини лабораторної роботи та перейду на неї. Для цього виконаю команди git branch та git checkout відповідно:



```
Git CMD
D:\KNU_FIT\3rd_grade\Programming technologies\Lab1\InfSec_MIT21>git branch test
D:\KNU_FIT\3rd_grade\Programming technologies\Lab1\InfSec_MIT21>git checkout test
Switched to branch 'test'
D:\KNU_FIT\3rd_grade\Programming technologies\Lab1\InfSec_MIT21>_
```

Нерідко виникає ситуація, коли потрібно перейти на іншу гілку, проте на поточній гілці містяться незбережені зміни, які поки що немає необхідності комітити. Тут стає в пригоді команда git stash. Вона дозволяє тимчасово приховати зміни, зроблені в поточній гілці, тим самим не вимагаючи виконання коміту. Згодом ці зміни можна буде дістати за допомогою команди git stash apply або git stash pop (ця команда не лише застосовує зміни, а й видаляє їх зі сховища, в якому вони тимчасово зберігалися).

Використання параметра -u з командою git stash дозволяє приховати невідстежувані Git-ом файли.

Зміню один із файлів у гілці test та приховую його:

```
Git CMD
D:\KNU_FIT\3rd_grade\Programming technologies\Lab1\InfSec_MIT21>git stash
Saved working directory and index state WIP on test: 22e001f Little changes to Lab 7-8

D:\KNU_FIT\3rd_grade\Programming technologies\Lab1\InfSec_MIT21>git status
On branch test
nothing to commit, working tree clean

D:\KNU_FIT\3rd_grade\Programming technologies\Lab1\InfSec_MIT21>git stash pop
On branch test
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   LAB1/task1/Program.cs

no changes added to commit (use "git add" and/or "git commit -a")
Dropped refs/stash@{0} (7cab04aefcbb5cd7241b36d93bc6d96b0851fc79)

D:\KNU_FIT\3rd_grade\Programming technologies\Lab1\InfSec_MIT21>git status
On branch test
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   LAB1/task1/Program.cs

no changes added to commit (use "git add" and/or "git commit -a")
D:\KNU_FIT\3rd_grade\Programming technologies\Lab1\InfSec_MIT21>
```

Як можна побачити, після приховування змін вивід команди `git status` свідчить про те, що репозиторій наразі чистий і не містить змін, які вимагали б нагального виконання коміту. Водночас після повернення змін командою `git stash pop` команда `git status` виявляє, що був змінений файл `Program.cs`.

Також є можливість очищення репозиторію шляхом видалення невідстежуваних файлів за допомогою команди `git clean`. Вона вимагає використання щонайменше одного параметру, такого як, наприклад, `-n` (показує, які файли буде видалено, якщо виконати цю команду, проте не видаляє їх фактично) або `-f` (власне видаляє файли, тому цей параметр слід використовувати обережно!).

Для демонстрації роботи цієї команди створю файл `test.txt` і видаляю його:

```
Git CMD
D:\KNU_FIT\3rd_grade\Programming technologies\Lab1\InfSec_MIT21>git clean -n
Would remove LAB1/task1/test.txt

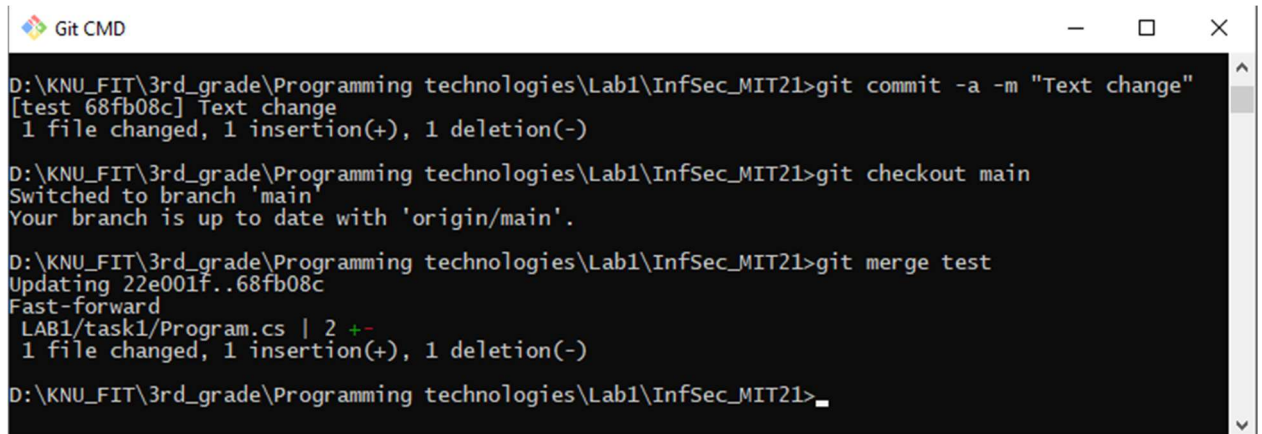
D:\KNU_FIT\3rd_grade\Programming technologies\Lab1\InfSec_MIT21>git clean -f
Removing LAB1/task1/test.txt

D:\KNU_FIT\3rd_grade\Programming technologies\Lab1\InfSec_MIT21>
```



Перша команда `git clean` лише попереджає про те, що буде видалено файл, в той час як друга безпосередньо видаляє його без можливості відновлення.

Зроблю коміт у гілці `test` для збереження змін, які приховувалися раніше, та виконаю злиття цієї гілки з гілкою `master` за допомогою команди `git merge`:



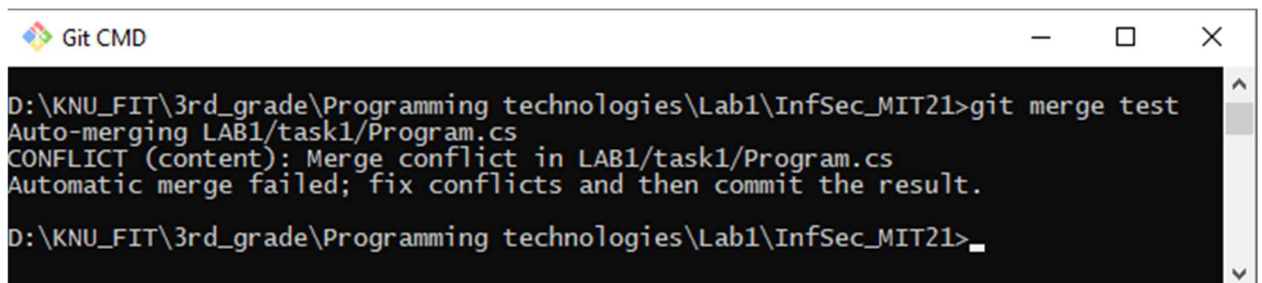
```
Git CMD
D:\KNU_FIT\3rd_grade\Programming technologies\Lab1\InfSec_MIT21>git commit -a -m "Text change"
[test 68fb08c] Text change
1 file changed, 1 insertion(+), 1 deletion(-)

D:\KNU_FIT\3rd_grade\Programming technologies\Lab1\InfSec_MIT21>git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

D:\KNU_FIT\3rd_grade\Programming technologies\Lab1\InfSec_MIT21>git merge test
Updating 22e001f..68fb08c
Fast-forward
 LAB1/task1/Program.cs | 2 +-
 1 file changed, 1 insertion(+), 1 deletion(-)

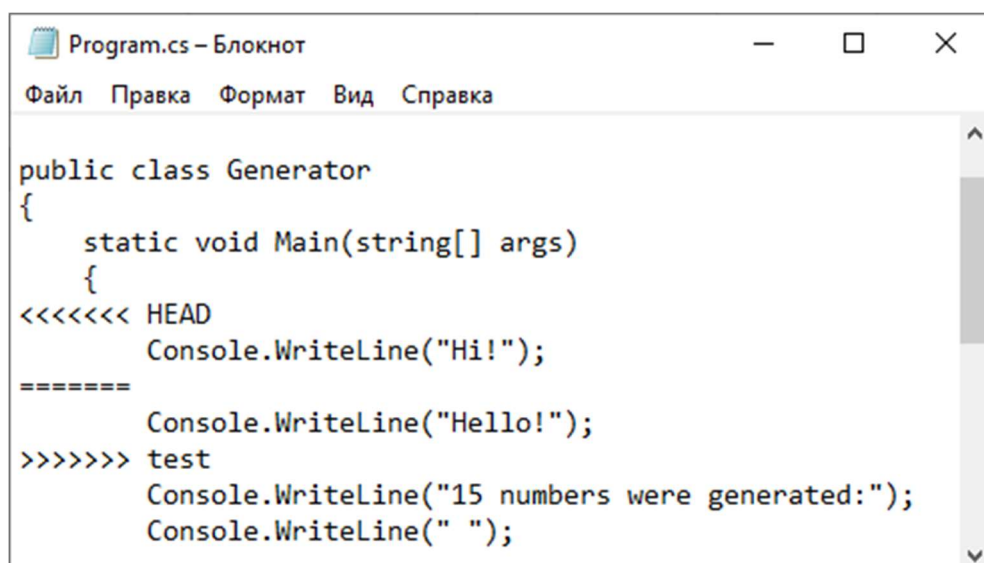
D:\KNU_FIT\3rd_grade\Programming technologies\Lab1\InfSec_MIT21>
```

Тепер зміню файл `Program.cs` в одному місці одночасно у двох гілках. Це призведе до конфлікту, відповідно `fast-forward` злиття не зможе виконатися і буде призупинене до моменту вирішення конфлікту:



```
Git CMD
D:\KNU_FIT\3rd_grade\Programming technologies\Lab1\InfSec_MIT21>git merge test
Auto-merging LAB1/task1/Program.cs
CONFLICT (content): Merge conflict in LAB1/task1/Program.cs
Automatic merge failed; fix conflicts and then commit the result.

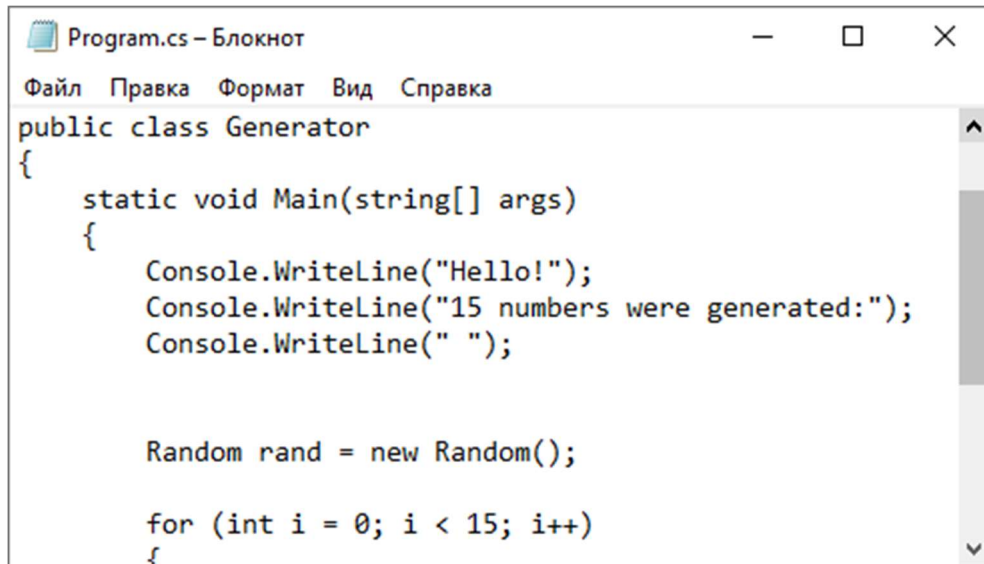
D:\KNU_FIT\3rd_grade\Programming technologies\Lab1\InfSec_MIT21>
```



```
Program.cs - Блокнот
Файл  Правка  Формат  Вид  Справка

public class Generator
{
    static void Main(string[] args)
    {
<<<<<<< HEAD
        Console.WriteLine("Hi!");
=====
        Console.WriteLine("Hello!");
>>>>>>> test
        Console.WriteLine("15 numbers were generated:");
        Console.WriteLine(" ");
    }
}
```

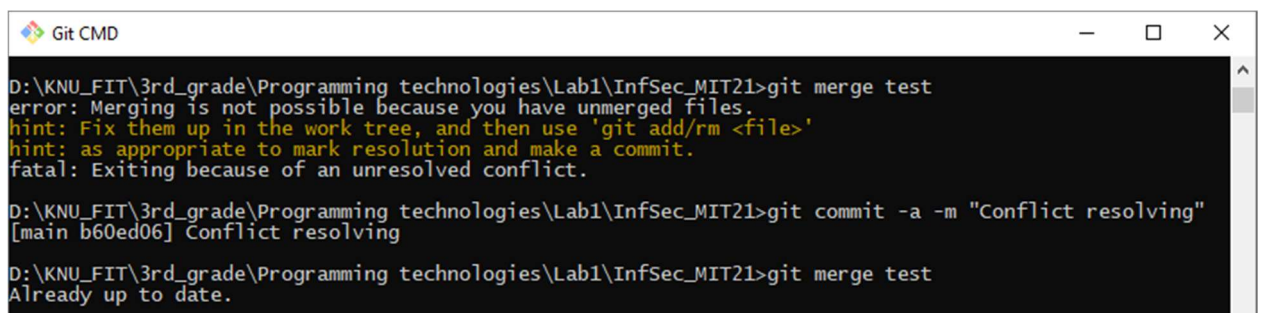
Вручну редагую файл і роблю коміт:



```
public class Generator
{
    static void Main(string[] args)
    {
        Console.WriteLine("Hello!");
        Console.WriteLine("15 numbers were generated:");
        Console.WriteLine(" ");

        Random rand = new Random();

        for (int i = 0; i < 15; i++)
        {
```



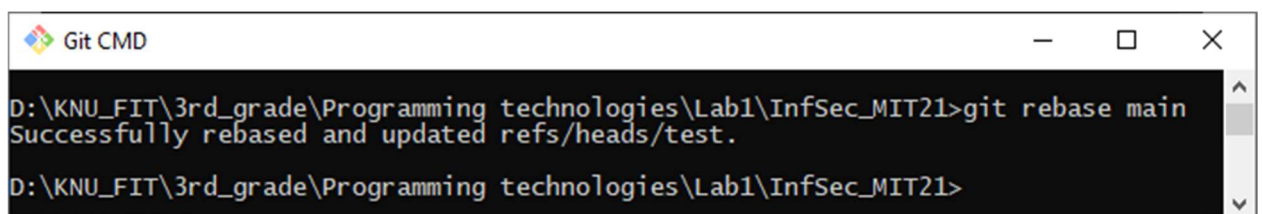
```
D:\KNU_FIT\3rd_grade\Programming technologies\Lab1\InfSec_MIT21>git merge test
error: Merging is not possible because you have unmerged files.
hint: Fix them up in the work tree, and then use 'git add/rm <file>'
hint: as appropriate to mark resolution and make a commit.
fatal: Exiting because of an unresolved conflict.

D:\KNU_FIT\3rd_grade\Programming technologies\Lab1\InfSec_MIT21>git commit -a -m "Conflict resolving"
[main b60ed06] Conflict resolving

D:\KNU_FIT\3rd_grade\Programming technologies\Lab1\InfSec_MIT21>git merge test
Already up to date.
```

Інтегрувати зміни з гілки test у гілку main можна також за допомогою переbazовування (rebase). Цей спосіб полягає у перенесенні комітів поточної гілки поверх комітів тієї, в яку виконується переbazовування. Іншими словами, коміти цільової гілки «відтісняються» комітами поточної і об'єднуються з ними, внаслідок чого зберігається чиста, а головне, зрозуміла історія комітів.

Переbazовування здійснюється командою git rebase:



```
D:\KNU_FIT\3rd_grade\Programming technologies\Lab1\InfSec_MIT21>git rebase main
Successfully rebased and updated refs/heads/test.

D:\KNU_FIT\3rd_grade\Programming technologies\Lab1\InfSec_MIT21>
```

**Висновок:** в ході виконання лабораторної роботи було розглянуто призначення та функціональні можливості системи контролю версій Git. Ми навчилися виконувати базові операції в Git, працювати з гілками і зливати їх між собою, а також з'ясували причини виникнення конфліктів та дізналися про способи їх оперативного вирішення.