**Київський національний університет імені Тараса Шевченка факультет радіофізики, електроніки та комп’ютерних систем**

Лабораторна робота № 1

**Тема: “Distributed version-control system Git**”

Роботу виконав студент 3 курсу

КІ-МА, ФРЕКС

Попов Олексій

Київ 2020

**Мета роботи:** ознайомитися системами керування версіями. Дослідити

та отримати практичні навики щодо створення найпростішої програми та

власного репозиторію.

**Завдання на лабораторну роботу:**

Ознайомитись з теоретичними відомостями, ретельно опрацювати

матеріал. Вміти давати пояснення термінам та поняттям: система керування

версіями; централізовані та розподілені системи контролю версіями;

репозиторій; приватні та відкриті репозиторії; GitHub.

**Хід виконання роботи:**

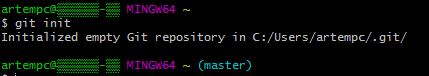
***Практична частина.***

*Настроювання Git.*

*1. Відкрийте папку, в якій знаходяться файли проекту. З контекстного*

*меню виберіть Git Bash Here , і у вікні, що відкрилося введіть команду:*

*$ git init*



*У результаті в папці з проектом з`являється папка з іменем «.git», що*

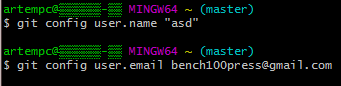
*містить усі необхідні файли локального репозиторію.*

*2. Авторизуйтесь для внесення подальших змін, вкажіть своє ім’я та*

*електронну пошту, наприклад:*

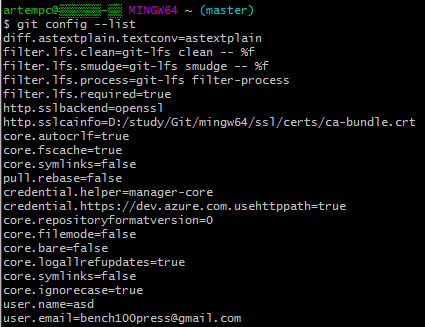
*$ git config user.name “asd”*

*$ git config user.email bench100press@gmail.com*



*Перевірити настройки можна за допомогою команди git config --list*

*Щоб отримати допомогу необхідно набрати git help назва команди*



3. Створіть в папці з проектом файл «.gitignore» для постійного

виключення з файлів зі змінами, що відстежуються, тимчасових файлів та

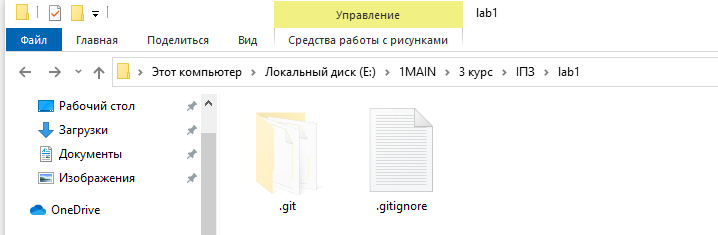
файлів, що автоматично генеруються компіляторам та компоновщиком.

При створенні файлу з іменем «.gitignore» за допомогою контекстного

меню можлива відмова системи Windows зберегти файл без вказування імені

перед точкою. В такому разі збережіть його з будь-яким ім’ям, наприклад,

«a.gitignore», а потім в командному рядку cmd або Git CMD перейменуйте його

за допомогою команди ren, попередньо перейшовши до поточної папки:

Перейменувати файл можна також за допомогою файлового менеджера.

Створіть у в файлі «.gitignore» список файлів та папок, які не потрібно

відслідковувати, наприклад:

\*~

\*obj3

Win32/

Debug/

\_\_history/

Символ «\*» заменює будь яку кількість символів, а символ «/» після

імені означає папку.

4. Імпортування папки в Git.

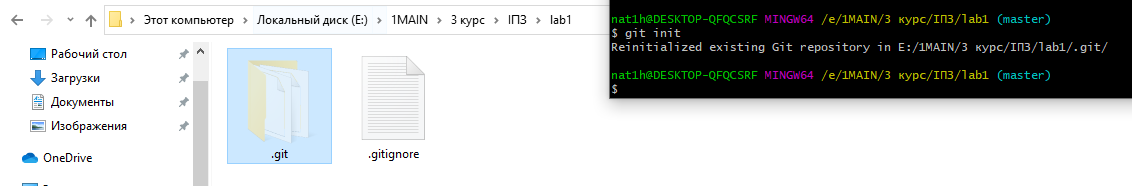
Необхідно зайти в поточну папку проекту і виконати команду

git init

В результаті виконання команди має з’явитись невидима папка [.git] зі

структурою робочих папок git, або повідомлення Reinitialized existing Git

repository in Path/Folder/.git/ ,якщо процес ініціалізації виконувався раніше.



В поточній папці проекту кожен файл може знаходитись в двох станах, під

вірсійним контролем і без нього, тобто коли файл не відслідковується

(untracked).

За допомогою команди git status можна подивитись в якому стані знаходиться

репозиторій.

Для того щоб файл почав відслідковуватись необхідно виконати команду

add <file>

Додайте всі файли, що знаходяться в папці для відслідковування за допомогою

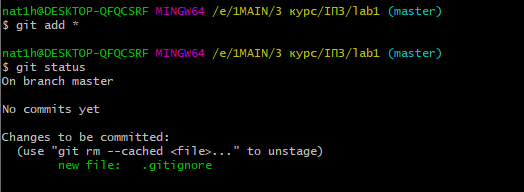
команди

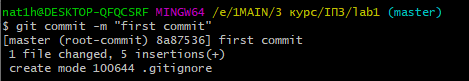
add \*

Виконайте команду git status і перевірте стан файлів.

Відредагуйте якийсь файл проекту, збережіть зміни і знову виконайте команду

git status

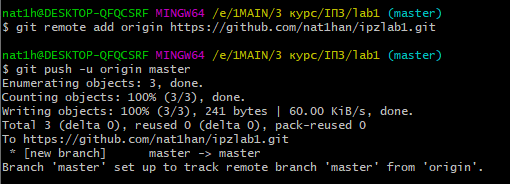




5. Додавання імен віддалених репозиторіїв.

Одним з найпопулярніших репозиторіїв є github. На ньому необхідно

зареєструватись і створити account.

Далі в консольному вікні необхідно виконати команду, наприклад:

Клонування репозиторію з сервера.

При спільній роботі над проектами кількох розробників в першу чергу

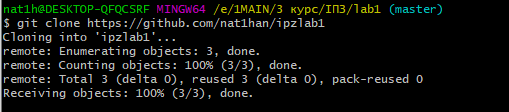
потрібно отримати актуальні версії проекту з віддаленого репозиторію. Для

цього необхідно знати коротке ім`я сервера, на якому знаходиться знаходиться

віддалений репозиторій або його URL-адреса. Знання короткого імені

достатньо, якщо файли в робочу папку вже копіювали з сервера за допомогою

команди команди git clone, наприклад:

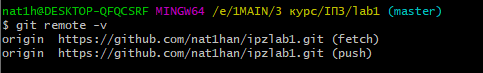


При цьому за замовчуванням віддаленому серверу буде присвоєне ім’я

«origin». Перевірити присвоєні імена серверів можна за допомогою команди git

remote :

$ git remote –v .



Ключ –v дає можливість вивести повні URL-адреси серверів: origin

https://github.com/ ім’я\_власника /ім’я\_папки (fetch) origin https://github.com/

ім’я\_власника /ім’я\_папки (push)

У дужках вказані команди скачування (fetch) і відправки змін (push)

допустимі для цього сервера.

Додати ще одне ім`я сервера з віддаленим репозиторієм можна за

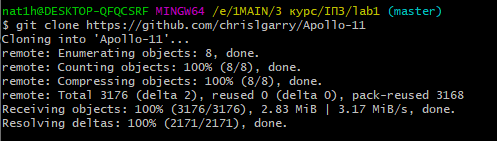
допомогою команди git remote add :

$ git remote add коротке\_ім’я\_сервера URL-репозиторію .

**Завдання**

Зробити клонування репозиторію коду космічної місії польоту на місяць.

<https://github.com/chrislgarry/Apollo-11>



При бажанні проглянути та перевірити програмні коди для комп’ютера

AGC (Apollo Guidance Computer) на симуляторі

<http://svtsim.com/moonjs/agc.html>.

**Основні команди Git**

5. Перевірка стану робочої папки здійснюється за допомогою команди

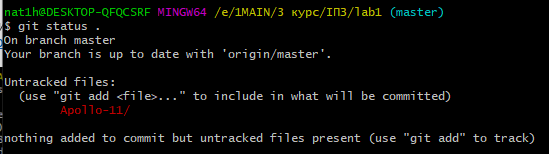
$ git status .

Git Bash покаже назву поточної гілки «master», присвоєне за

замовчуванням, і список усіх файлів в робочій папці, які не відслідковуються і

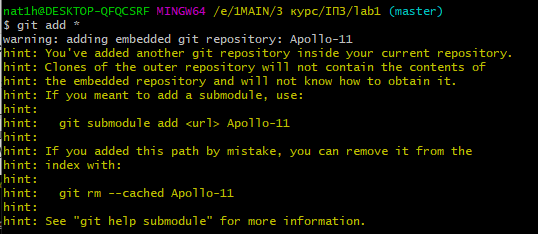
не підпадають під шаблони файлу «.gitignore». Імена цих файлів будуть

виділені червоним шрифтом.



6. Включення файлів і папок в перелік, що відслідковується виконується

командою git add:



$ git add імена\_файлів\_або\_шаблони\_імен .

Наприклад, команда

$ git add \*cpp \*dfm \*h .

включає в перелік файлів, що відслідковуються файли вихідних текстів

на мові C++ (\*cpp), опис графічної форми (\*dfm) і файли заголовків (\*h).

Внесіть зміни в один з текстових файлів, наприклад, додайте коментар до

якого-небудь оператора вихідного тексту програми на мові C ++, і збережіть

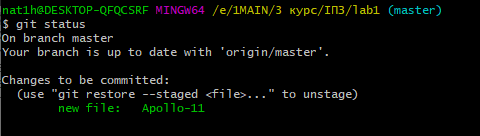
змінений файл. Знову виконайте команду git status. Перелік файлів робочої

папки, що відслідковуються, які не були змінені і підготовлені до фіксації їх

стану командою commit буде виведений зеленим шрифтом. Ім`я зміненого

файлу і файлів, які не включені в ті, що відслідковуються, будуть показані

червоним шрифтом.



7. Додайте змінений файл до файлів, що підготовлені до фіксації

стану (staged files), за допомогою команди

$ git add ім’я\_файла .

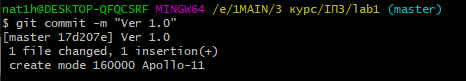
Перевірте стан файлів командою git status і збережіть знімок поточного

стану відслідковуються файлів командою git commit: $ git commit -m ‘Ver 1.0’.

Перевірте стан файлів за допомогою команди git status і збережіть знімок

поточного стану файлів, що відслідковуються за допомогою команди git

commit: $ git commit -m ‘Ver 1.0’.



Ключ «-m» дає можливість зберегти повідомлення (message)’ Ver 1.0’ як

коментар, що описує дану фіксацію (знімок стану). Повідомлення має містити

простий текст, поміщений в апострофи або подвійні лапки. В цьому прикладі

пропонується вказати, що цей знімок стану є першою версією файлів проекту.

Будь-яка фіксація потребує обов`язкового зазначення рядка повідомлення.

Введення команди git commit без вказівки повідомлення запускає

текстовий редактор, яким за замовчуванням є стандартний редактор тексту ОС

UNIX / Linux Vim. Редактор Vim завжди запускається в командному режимі.

Щоб перевести його в режим вставки, тобто в режим введення і зміни тексту,

необхідно ввести з клавіатури символи «i» або «a».

В першому пустому рядку відкритого тексту-коментаря необхідно ввести

повідомлення, яке описує дану фіксацію, a потім перейти в командний режим

натисканням клавіші <Esc>Вихід з редактора зі збереженням змін виконується

за допомогою команди

***:wq*** або ***ZZ .***

При цьому програма git виконає фіксацію стану файлів проекту зі

збереженням в якості коментаря рядка введеного повідомлення.

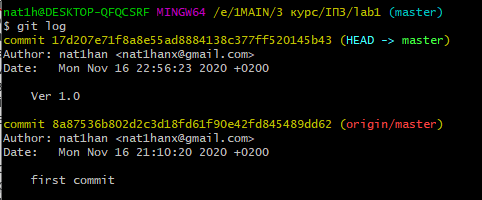
9. Перегляньте історію змін за допомогою команди git log:

$ git log або $ git log --pretty=oneline.

Ключ «--oneline» зменшує обсяг виведеної інформації. Ключ «-p» дає

можливість вивести відмінність між кожним commit, а ключ «число» - обмежує

виведення кількістю commit, заданих числом.



10. Створення нової гілки

Необхідність створення нової гілки в розробці проекту виникає, якщо

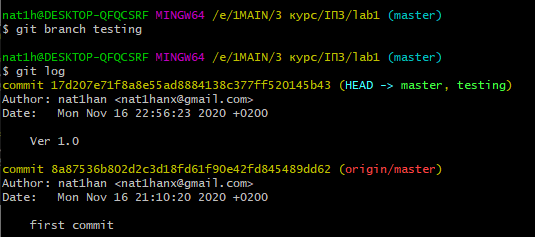
передбачаються суттєві відхилення від початкового проекту або розробникам

потрібне схвалення результатів роботи керівником проекту.

Створіть гілку testing за допомогою команди

$ git branch testing .

Дізнатись, яка гілка є поточною можна за допомого команди git log:



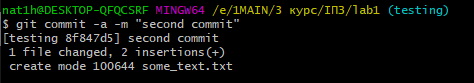
Перехід на нову гілку виконується за допомогою команди git checkout.



11. Внесіть зміни в один-два текстові файли нової гілки, що

відслідковуються і зафіксуйте зміни за допомогою команди

$ git commit -a -m “текст\_комментаря\_commit”



Ключ «-a» в команді git commit дає можливість пропустити команду git

add, що додає змінені файли в область індексуваня, та готує їх до фіксації.

Перегляньте історію змін гілки testing за допомогою команди git log --

oneline .

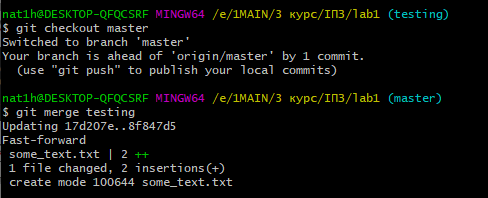
12. Об’єднування гілок.

Перейдіть на головну гілку master за допомогою команди

$ git checkout master .

Виконайте об’єднування гілки testing з гілкою master за допомогою

команди $ git merge testing.



Перегляньте історію змін за допомогою команди $ git log --graph --oneline

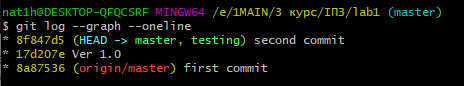
Ключ «--graph» дає можливість графічно показати історію розділення та

об’єднання гілок master и testing:

$ git log --graph --oneline

\* ed802bc (HEAD -&gt; master, testing) second commit

\* 9d7386f (origin/master) first commit

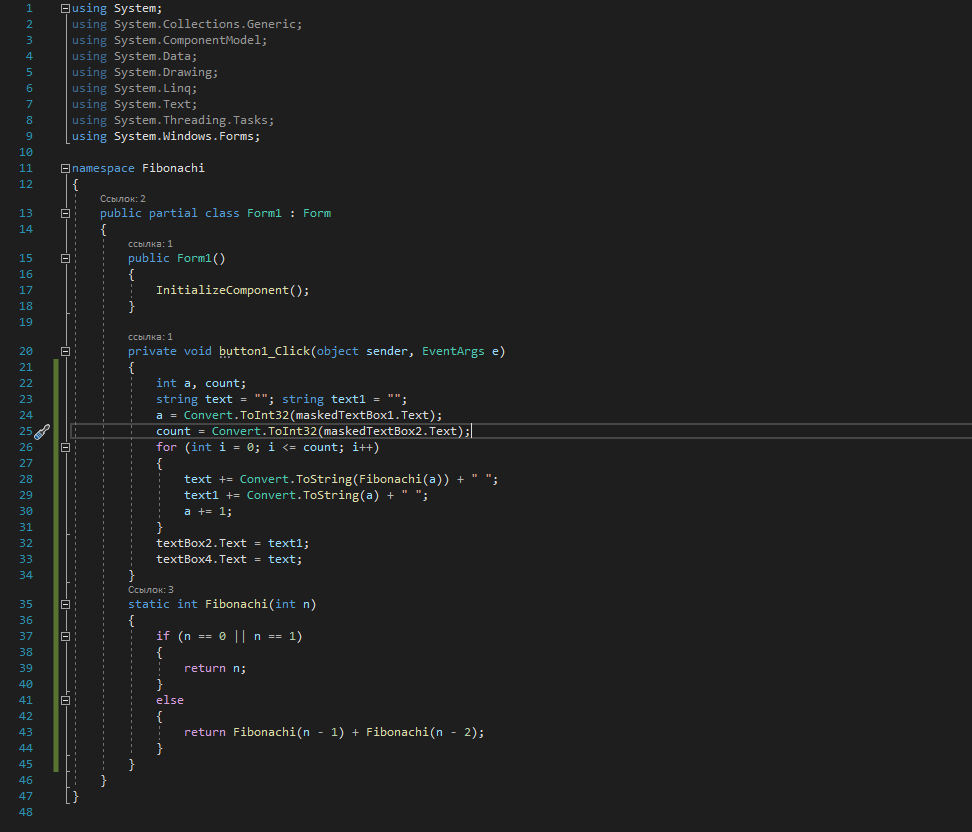
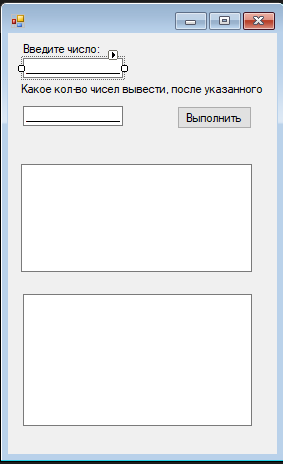


Перегляньте результат об’єднання виправлених файлів в текстовому

редакторі.

**Завдання 2.**

Написати програму для розрахунку чисел Фібоначчі та завантажити на Github.

1. Мовою C# написано програму для рахування та виведення ряду чисел Фібоначчі:

Код проекту:

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Fibonachi

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

private void button1\_Click(object sender, EventArgs e)

{

int a, count;

string text = ""; string text1 = "";

a = Convert.ToInt32(maskedTextBox1.Text);

count = Convert.ToInt32(maskedTextBox2.Text);

for (int i = 0; i <= count; i++)

{

text += Convert.ToString(Fibonachi(a)) + " ";

text1 += Convert.ToString(a) + " ";

a += 1;

}

textBox2.Text = text1;

textBox4.Text = text;

}

static int Fibonachi(int n)

{

if (n == 0 || n == 1)

{

return n;

}

else

{

return Fibonachi(n - 1) + Fibonachi(n - 2);

}

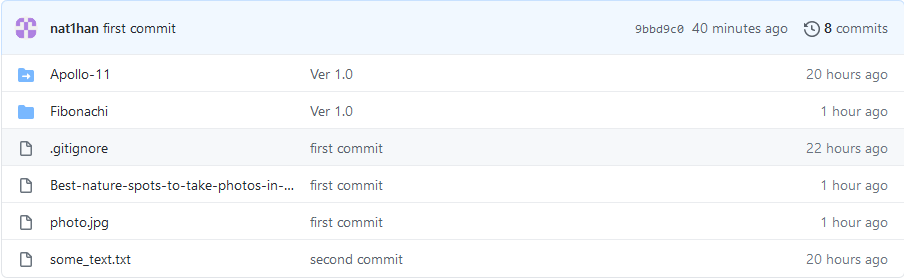
}

}

}

За допомогою цих команд завантажимо проект на GitHub:



Впевнимося, що все завантажено:

Висновки: В даній лабораторній роботі було освоєно навички роботи з розподіленими системами керування версіями на прикладі Git. Було навчено виконанню основних команд для роботи з Git. В результаті виконання цієї лабораторної роботи було показано зміст розподілених систем керування версіями. Також до репозиторію додано файл з кодом для рахування та виведення ряду чисел Фібоначчі

My github: https://github.com/nat1han/ipzlab1/