

ЛАБОРАТОРНА РОБОТА № 8

ЗНАЙОМСТВО З СИСТЕМОЮ КОНТРОЛЮ ВЕРСІЙ GIT.

Мета роботи: ознайомитися системами керування версіями. Дослідити та отримати практичні навички щодо створення найпростішої програми та власного репозиторію.

1. Теоретичні відомості

Система керування версіями (англ. source code management, SCM) — програмний інструмент для керування версіями одиниці інформації: вихідного коду програми, скрипту, веб-сторінки, веб-сайту, 3D моделі, текстового документу тощо.

Система керування версіями — це потужний інструмент, який дозволяє одночасно, без завад один одному, проводити роботу над груповими проектами.

Системи керування версіями зазвичай використовуються при розробці програмного забезпечення для відстеження, документування та контролю над поступовими змінами в електронних документах: у програмному коді застосунків, кресленнях, електронних моделях та інших документах, над змінами яких одночасно працюють декілька людей.

Кожна версія позначається унікальною цифрою чи літерою, зміни документу занотовуються. Зазвичай також зберігається автор зробленої зміни та її час.

Інструменти для контролю версій входять до складу багатьох інтегрованих середовищ розробки.

Система керування версіями існують двох основних типів: з централізованим сховищем та розподіленим (рис. 1).

Система збереження історії редагувань статей, що застосовується у Вікіпедії є прикладом системи керування версіями.

Система контролю дозволяє зберігати попередні версії файлів та завантажувати їх за потребою. Вона зберігає повну інформацію про версію кожного з файлів, а також повну структуру проекту на всіх стадіях розробки. Місце зберігання даних файлів називають репозиторієм. В середині кожного з репозиторіїв можуть бути створені паралельні лінії розробки — гілки.

Гілки зазвичай використовують для зберігання експериментальних, незавершених (alpha, beta) та повністю робочих версій проекту (final). Більшість систем контролю версії дозволяють кожному з об'єктів присвоювати теги, за допомогою яких можна формувати нові гілки та репозиторії.

Централізовані системи контролю версій

Централізована система контролю версій (клієнт-серверна) — система, дані в якій зберігаються в єдиному «серверному» сховищі. Весь обмін файлами відбувається з використанням центрального сервера. Є можливість створення та роботи з локальними репозиторіями (робочими копіями).

Переваги:

- Загальна нумерація версій;
- Дані знаходяться на одному сервері;
- Можлива реалізація функції блокування файлів;
- Можливість керування доступом до файлів;

Недоліки:

- Потреба в мережевому з'єднанні для оновлення робочої копії чи збереження змін;

До таких систем відносять Subversion, Team Foundation Server.

Розподілені системи контролю версій

Розподілена система контролю версій (англ. Distributed Version Control System, DVCS) — система, яка використовує замість моделі клієнт-сервер, розподілену модель зберігання файлів. Така система не потребує сервера, адже всі файли знаходяться на кожному з комп'ютерів.

Переваги:

- Кожний з розробників працює зі своїм власним репозиторієм;
- Рішення щодо злиття гілок приймається керівником проекту;
- Немає потреби в мережевому з'єднанні;

Недоліки:

- Немає можливості контролю доступу до файлів;
- Відсутня загальна нумерація версій файла;
- Значно більша кількість необхідного дискового простору;
- Немає можливості блокування файлів;

До таких систем відносять Git, Mercurial, SVK, Monotone, Codeville, BitKeeper

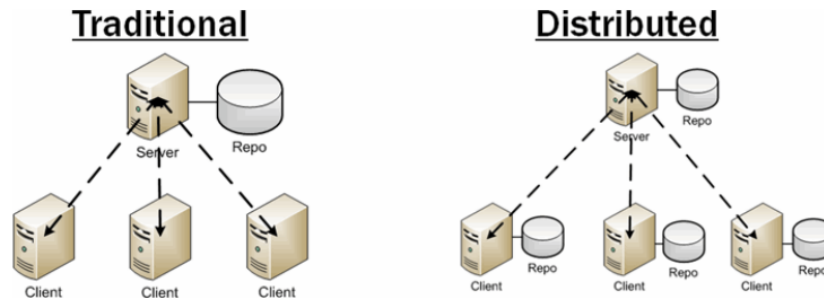


Рисунок 1 – Системи контролю версій

Використання системи контролю версій є необхідним для роботи над великими проектами, над якими одночасно працює велика кількість розробників. Системи контролю версій надають ряд додаткових можливостей:

- можливість створення різних варіантів одного документу;
- документування всіх змін (коли ким було змінено/додано, хто який рядок змінив);
- функція контролю доступу користувачів до файлів (є можливість його обмеження для різних користувачів);
- створення документації проекту з поетапним записом змін в залежності від версії;
- давання пояснення до змін та документування їх.

Найбільш відомими веб-сервісами для хостингу проектів на базі систем керування версіями є:

- GitHub (<https://github.com/>);
- GitLab (<https://gitlab.com/> / <https://gitlab.mircloud.us/>);
- BitBucket (<https://bitbucket.org/>);

GitHub — один з найбільших веб-сервісів для спільної розробки програмного забезпечення. Існують безкоштовні та платні тарифні плани користування сайтом. Базується на системі керування версіями Git і розроблений на Ruby on Rails і Erlang компанією GitHub, Inc (раніше Logical Awesome).

Розробники сайту називають GitHub «соціальною мережею для розробників».

Окрім розміщення коду, учасники можуть спілкуватись, коментувати

редагування один одного, а також слідкувати за новинами знайомих. За допомогою широких можливостей Git програмісти можуть поєднувати свої репозиторії – GitHub дає зручний інтерфейс для цього і може показувати вклад кожного учасника в вигляді дерева.

Для проектів є особисті сторінки, невеликі Вікі та система відстеження помилок. Прямо на сайті можна дивитись файли проектів з підсвічуванням синтаксису для більшості мов програмування.

Кількість приватних (закритих для перегляду користувачами Інтернету) репозиторіїв – 5. Для того, щоб мати можливість створювати більше приватних репозиторіїв потрібно переходити на платний тарифний план. Кількість відкритих репозиторіїв – необмежена.

GitLab — сайт та система керування репозиторіями програмного коду для Git, з додаткових можливостей: власна вікі та система відстеження помилок. GitLab — компанія, що пропонує схожі з GitHub користувацькі послуги із додатковими перевагами, як то приватні репозиторії для безкоштовних підписників. Також суттєвою перевагою є можливість розгорнути систему на сторонніх серверах. Програмне забезпечення для GitLab була написана Валерієм Сизовим з України.

Завдання на лабораторну роботу:

1. 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; GitLab.

2. Зареєструватися на сайті GitLab та створити репозиторій:

2.1. Зареєструватися на сайті <https://gitlab.com>


GitLab.com

GitLab.com offers free unlimited (private) repositories and unlimited collaborators.

- [Explore projects on GitLab.com](#) (no login needed)
- [More information about GitLab.com](#)
- [GitLab.com Support Forum](#)
- [GitLab Homepage](#)

By signing up for and by signing in to this service you accept our:

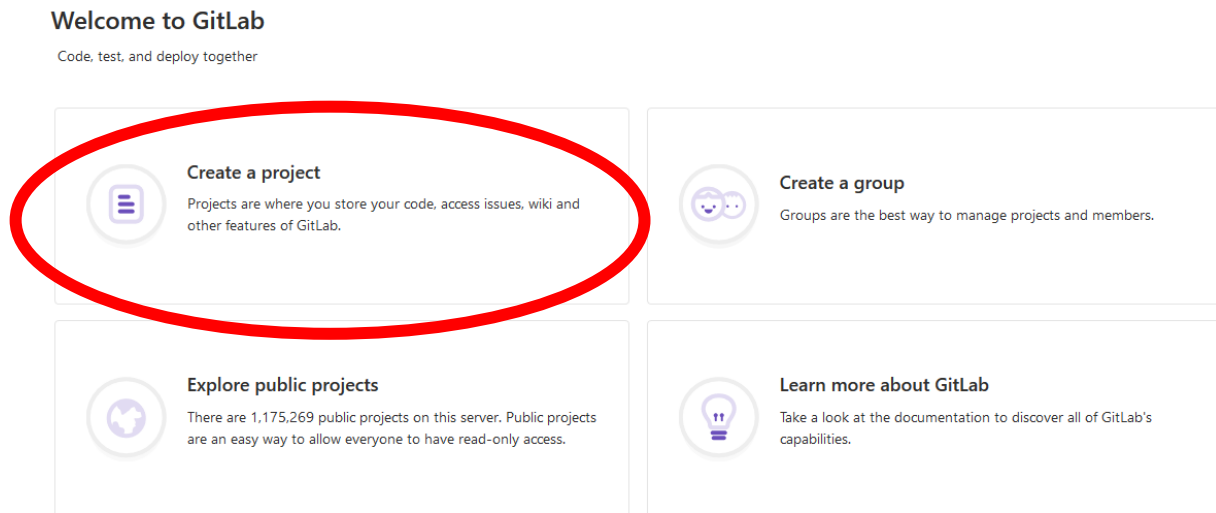
- [Privacy policy](#)
- [GitLab.com Terms](#).

Sign in	Register
Full name	
<input type="text"/>	
Username	
<input type="text"/>	
Email	
<input type="text"/>	
Email confirmation	
<input type="text"/>	
Password	
<input type="text"/>	
Minimum length is 8 characters	
<input type="checkbox"/> I accept the Terms of Service and Privacy Policy	
<input type="checkbox"/> I'd like to receive updates via email about GitLab.	
<input type="checkbox"/> Я не робот	
	
<small>Конфідційність • Умови використання</small>	
<input type="button" value="Register"/>	

2.2. Увійти на власну пошту та підтвердити реєстрацію у листі, який надійшов з сервера GitLab.

2.3. Повернутися на сайт GitLab і увійти під власним логіном та паролем.

2.4. Створити перший репозиторій з назвою «TCPP-Lab8»:



Project name: TCPP-Lab8

Project description: Лабораторна робота №8 з ООП. Знайомство з системою контролю версій Git.

Visibility Level: Private

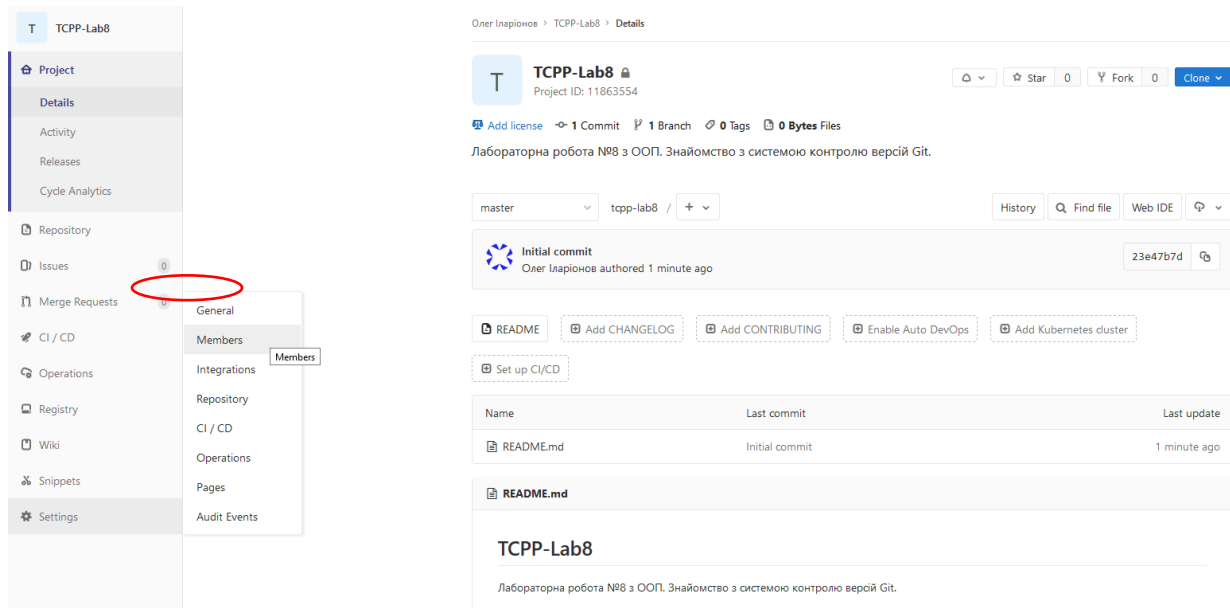
The image shows the 'New project' form in GitLab. On the left, there is a 'New project' section with instructions and a tip. The main form area has four tabs: 'Blank project' (selected), 'Create from template', 'Import project', and 'CI/CD for external repo'. The 'Blank project' tab contains fields for 'Project name' (filled with 'TCPP-Lab8'), 'Project URL' (filled with 'https://gitlab.com/oleg.ilarionov/'), and 'Project slug' (filled with 'tcp-lab8'). There is also a 'Project description (optional)' text area filled with 'Лабораторна робота №8 з ООП. Знайомство з системою контролю версій Git.' Below these fields is the 'Visibility Level' section with three radio buttons: 'Private' (selected), 'Internal', and 'Public'. At the bottom, there is a checkbox for 'Initialize repository with a README' and a green 'Create project' button.

2.5. Надати доступ до репозиторію викладачам:

- Іларіонову О.Є. (oleg.ilarionov@knu.ua)

- Гамоцькій С.Л. (_____)

Для цього перейти до розділу «Members»:



Ввести електронну пошту викладача «oleg.ilarionov@knu.ua», вибрати зі списку знайденого користувача:

Поставити рівень доступу: «Maintainer»

Choose a role permission

Maintainer

[Read more about role permissions](#)

Access expiration date

Expiration date

[Add to project](#) [Import](#)

Existing members and groups

Members of TCP-PP-Lab8

Find existing members by name

Name, ascending

Олександр Іларіонов @oleg.ilarionov [It's you](#)

Given access 3 minutes ago

Maintainer

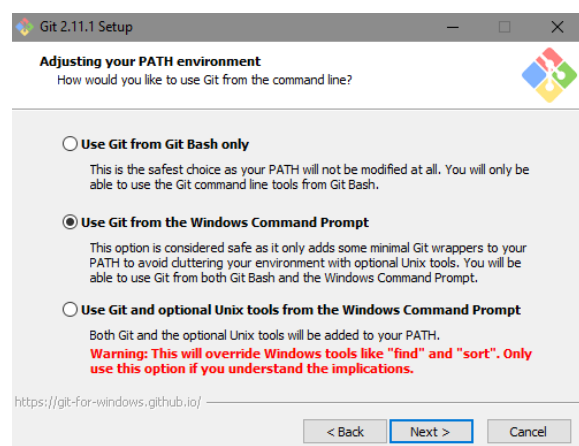
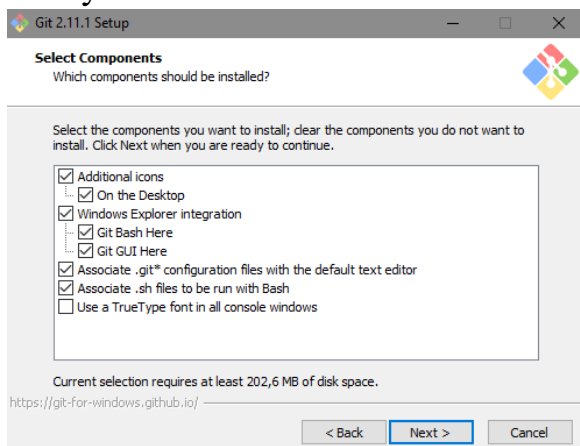
Натиснути «Add to project».

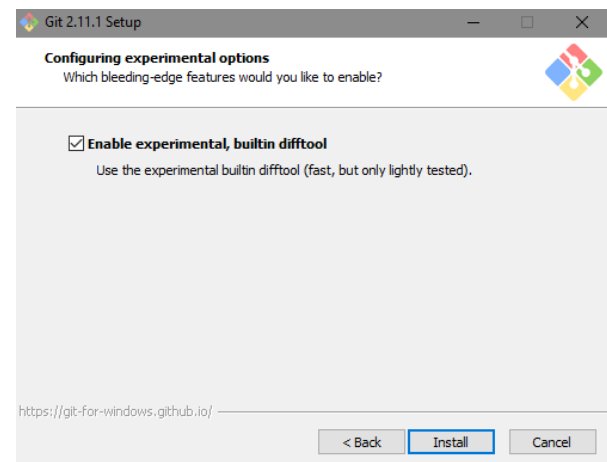
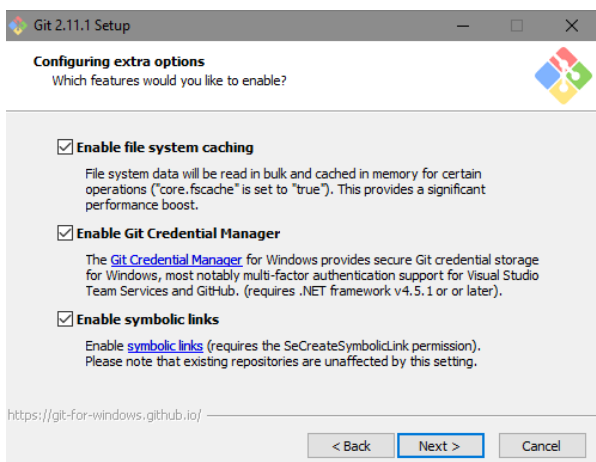
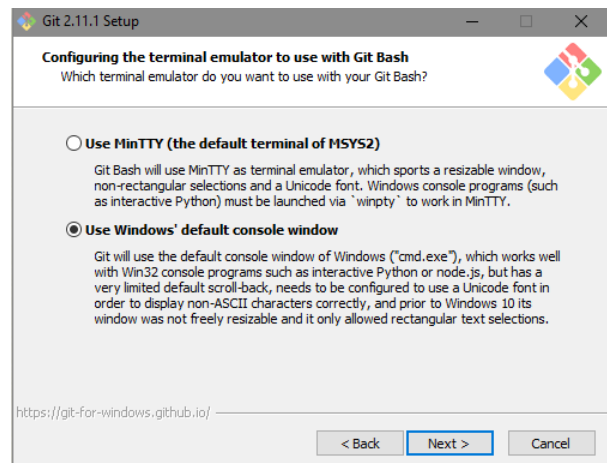
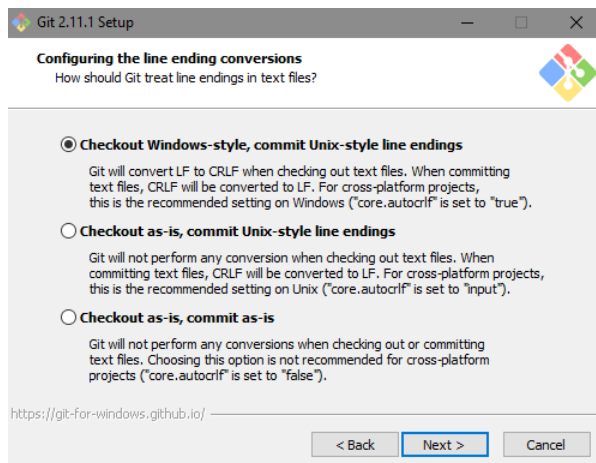
3. Встановити програмне забезпечення для системи контролю версій

3.1. Зайти на сайт (<https://git-scm.com/>), скачати установочний файл.

3.2. Запустити установочний файл і у діалогових вікнах вибрати такі

налаштування:





4. Створення проекту у Visual Studio.

4.1. Створити консольний проект Visual C++ з такими параметрами:

- Назва рішення: TCPP-Lab8;
- Назва проекту: Lab8.

4.2. У програму додати два рядки:

```
#include "pch.h"
#include <iostream>

int main()
{
    std::cout << "Hello World!\n";
}
```

5. Виконати початкове налаштування локального репозиторію.

5.1. Відкрити командний рядок (комбінація клавіш Win-R, команда "cmd").

5.2. Виконуємо початкові налаштування:

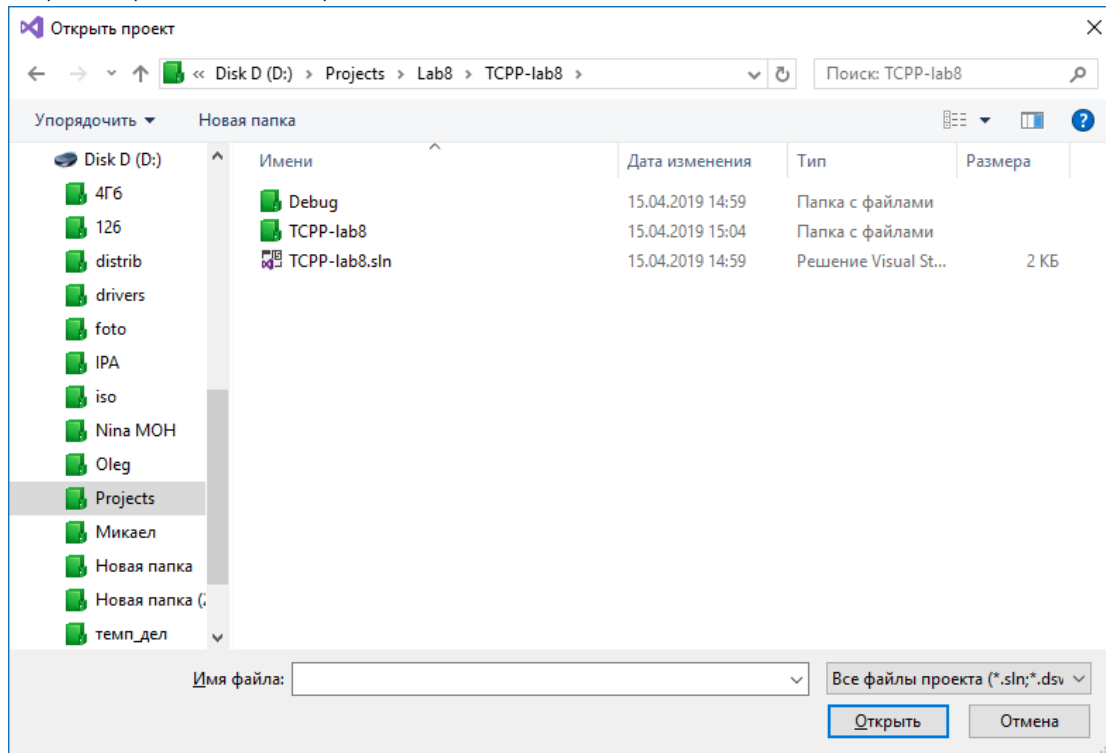
```
git config --global user.name "ilarionov"
git config --global user.email "oleg.ilarionov@knu.ua"
```

Ці дії потрібно виконати лише один раз, після встановлення Git на комп'ютер.

4.3. Перейти у каталог, де зберігається папка рішення, використовуючи

команди командного рядка Windows.

Наприклад, якщо рішення зберігається на диску D у папці Projects\Lab8\TCPP-Lab8\:



Тоді потрібно виконати дві команди:

```
d:  
cd Projects\Lab8\TCPP-lab8 \
```

Результат виконання:

```
C:\Windows\system32\cmd.exe  
  
c:\>d:  
  
D:\>cd Projects\Lab8\TCPP-lab8\TCPP-lab8  
  
D:\Projects\Lab8\TCPP-lab8\TCPP-lab8>
```

Звертаємо увагу, що потрібно перейти саме до папки з рішенням, а не окремим проектом.

4.4. Виконуємо ініціалізацію локального репозиторію для поточного рішення. Цю дію потрібно виконати один раз для нового рішення:

```
git init
```

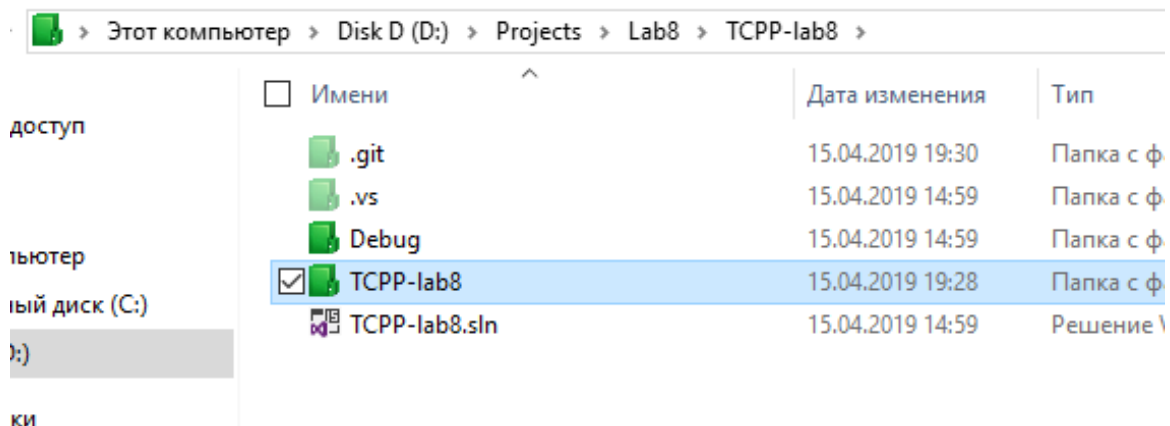
Результат роботи команди:

```
C:\Windows\system32\cmd.exe

D:\Projects\Lab8\TCPP-lab8>git init
Initialized empty Git repository in D:/Projects/Lab8/TCPP-lab8/.git/

D:\Projects\Lab8\TCPP-lab8>_
```

Повинен з'явитися службовий каталог .git, в якому будуть зберігатися службові файли репозиторію. Перевіримо, чи створився він:



5.5. Переглянемо статус локального репозиторію:

```
git status
```

Результат роботи команди:

```
C:\Windows\system32\cmd.exe

D:\Projects\Lab8\TCPP-lab8>git status
On branch master

No commits yet

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

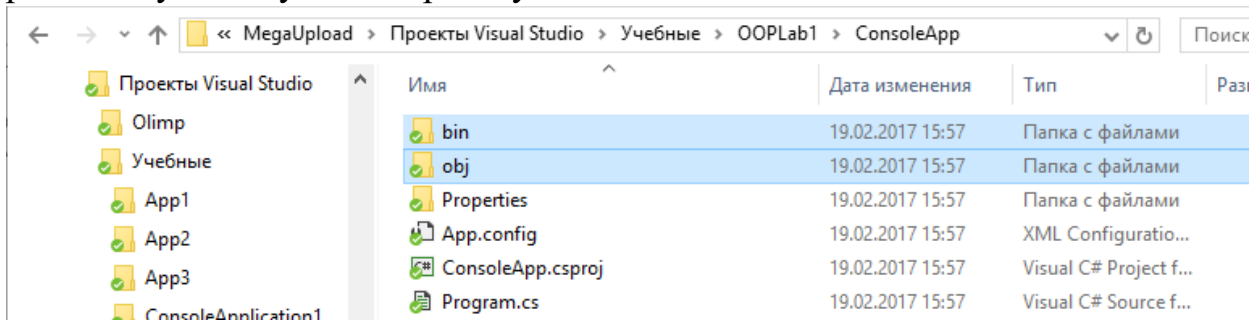
        .vs/
        Debug/
        TCPP-lab8.sln
        TCPP-lab8/

nothing added to commit but untracked files present (use "git add" to track)
D:\Projects\Lab8\TCPP-lab8>_
```

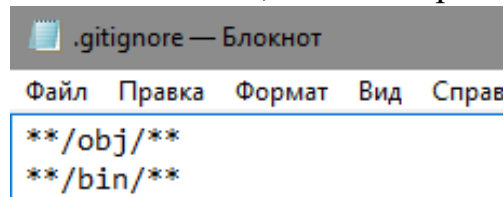
4.6. Тепер потрібно додати файли, які будуть індексуватися у git-репозиторії.

Однак, не всі файли потрібно буде завантажувати у віддалений

репозиторій, який зберігатиметься на сайті. Зокрема, не потрібно завантажувати файли, отримані в результаті компіляції проекту. Це папки «obj», «bin», які розташовуються у папці проекту:



Створимо файл `.gitignore`, який буде розміщуватись у папці рішення і міститиме два рядки з назвами каталогів, які не потрібно буде індексувати.



Для того, щоб створити цей файл виконаємо дві команди:

```
echo **/obj/**> .gitignore
echo **/bin/**> .gitignore
```

«**» Означає будь-які файли. Тобто з індексування виключаються папки «obj» та «bin», які розміщуються у будь-якому підкаталозі та які містять будь-які файли.

4.7. Тепер можна виконати додавання файлів до індексування. Для цього виконуємо команду:

```
git add *.*
```

4.8. Переглядаємо файли, які було додано до індексування:

```
git status
```

Результат виконання команди:

```
C:\Windows\system32\cmd.exe

D:\Projects\Lab8\TCPP-lab8>git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)

        new file:   .vs/TCPP-lab8/v15/.suo
        new file:   .vs/TCPP-lab8/v15/Browse.VC.db
        new file:   .vs/TCPP-lab8/v15/ipch/a48f2ff185122d0b.ipch
        new file:   Debug/TCPP-lab8.exe
        new file:   Debug/TCPP-lab8.ilc
        new file:   Debug/TCPP-lab8.pdb
        new file:   TCPP-lab8.sln
        new file:   TCPP-lab8/Debug/TCPP-lab8.log
        new file:   TCPP-lab8/Debug/TCPP-lab8.obj
        new file:   TCPP-lab8/Debug/TCPP-lab8.pch
        new file:   TCPP-lab8/Debug/TCPP-lab8.tlog/CL.command.1.tlog
        new file:   TCPP-lab8/Debug/TCPP-lab8.tlog/CL.read.1.tlog
        new file:   TCPP-lab8/Debug/TCPP-lab8.tlog/CL.write.1.tlog
        new file:   TCPP-lab8/Debug/TCPP-lab8.tlog/TCPP-lab8.lastbuildstate
        new file:   TCPP-lab8/Debug/TCPP-lab8.tlog/link.command.1.tlog
        new file:   TCPP-lab8/Debug/TCPP-lab8.tlog/link.read.1.tlog
        new file:   TCPP-lab8/Debug/TCPP-lab8.tlog/link.write.1.tlog
        new file:   TCPP-lab8/Debug/pch.obj
        new file:   TCPP-lab8/Debug/vc141.idb
        new file:   TCPP-lab8/Debug/vc141.pdb
        new file:   TCPP-lab8/TCPP-lab8.cpp
        new file:   TCPP-lab8/TCPP-lab8.vcxproj
        new file:   TCPP-lab8/TCPP-lab8.vcxproj.filters
        new file:   TCPP-lab8/TCPP-lab8.vcxproj.user
        new file:   TCPP-lab8/pch.cpp
        new file:   TCPP-lab8/pch.h
```

4.9. Тепер зафіксуємо поточний стан цих файлів. Операції фіксації змін називається комітом. Для того, щоб зробити коміт потрібно виконати команду:

```
git commit -m "Створено найпростіший консольний додаток"
```

Обов'язково додавайте зрозумілий і розширений опис для комітів.

4.10. Подивимось список комітів з поясненнями виконавши команду:

```
git log
```

Результат виконання команди:

```
C:\Windows\system32\cmd.exe

D:\Projects\Lab8\TCPP-lab8>git log
commit 84e31462fa6c5574db1b50c8539bc7caee120afe (HEAD -> master)
Author: ilarionov <oilarionov@gmail.com>
Date:   Mon Apr 15 19:57:30 2019 +0300

    <D0><A1><D1><82><D0><B2><D0><BE><D1><80><D0><B5><D0><BD><D0><BD><D0><BE><D0><BD><D0><B0><D0><B9><D0><BF><D1><80><D0><BE><D1><81><D1><82><D1><96><D1><88><D0><B8><D0><B9><D0><BA><D0><BE><D0><BD><D1><81><D0><BE><D0><BB><D1><8C><D0><BD><D0><B8><D0><B9><D0><B4><D0><BE><D0><B4><D0><B0><D1><82><D0><BE><D0><BA>

D:\Projects\Lab8\TCPP-lab8>
```

5. Додавання змін до рішення та завантаження їх на сервер

5.1. Ввести зміни у програмний код консольного додаток та щоб можна було знаходити значення функції (функція задається викладачем)

Приклади функцій

1. $f(x) = 15 + \frac{x}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!}$
2. $f(x) = 4.5 \cos \frac{3.33x}{0.25} + 2.1 \sin \frac{2.2x}{1.1}$
3. $f(x) = \pi + 2.1 \cdot x^2 + e^x + 0.38 \cdot x^3$

5.2. Завантажити етапи створення зміненого додатку у вигляді комітів на сервер Git. Для того, щоб коміти можна було завантажити на сервер Git виконуємо команди:

```
git add *.*
git add -u
git commit -m "Додано підрахунок суми чисел і виведення результату"
```

Перша команда визначає, які з файлів рішення були змінені. Друга команда фіксує ці зміни і готує коміт.

6. Підключення віддаленого репозиторію та відправка комітів

6.1. Переглядаємо список віддалених репозиторіїв, які вже підключені у вашій системі:

```
git remote
```

Якщо команда нічого не вивела, то це означає, що віддалені репозиторії у вашій системі ще не налаштовувались.

Але якщо команда вивела список репозиторіїв, то потрібно від'єднатися від них.

Наприклад, якщо результат роботи команди такий:

```
D:\MegaUpload\Проекты Visual Studio\Учебные\ООPLab1>git remote
origin
```

```
User@PC_Home15 MINGW64 /d/Projects/Lab8/TCPP-lab8 (master)
$ git remote
origin
```

```
User@PC_Home15 MINGW64 /d/Projects/Lab8/TCPP-lab8 (master)
$ _
```

то потрібно виконати команду для кожного віддаленого репозиторію зі списку:

```
git remote remove origin
```

6.2. Додаємо зв'язок з віддаленим репозиторієм на GitLab. Для цього виконуємо команду:

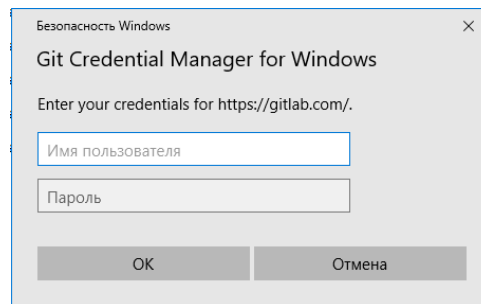
```
git remote add origin https://gitlab.com/oleg.ilarionov/tcpp-lab08.git
```

Адресу репозиторію можна подивитися на сайті GitLab.

6.3. Завантажити підготовлені локальні коміти на сервер.

```
git push -u origin master
```

6.4. З'явиться вікно, у якому потрібно буде ввести логін та пароль від сайту GitLab:



6.5. Якщо всі дії виконано правильно, то з'явиться повідомлення про успішне завантаження файлів.

Якщо ж логін та пароль введено не правильно, то отримаємо повідомлення «remote: HTTPBasic: Accessdenied»:

Якщо виникає помилка виду:

```
C:\Users\ki2_kv1\Documents\Visual Studio 2013\Projects\OOP1Lab1>git push -u origin master
fatal: unable to access 'https://gitlab.com/ki2_kv1/oop-lab1.git/': error setting certificate verify locations:
  CAfile: C:/Users/pi54_vpp/AppData/Local/Programs/Git/mingw64/ssl/certs/ca-bundle.crt
  CApath: none
```

то виконайте команду:



```
git config --system http.sslverify false
```





А потім:






```
git push -u origin master
```


6.6. Переконаємось, що всі файли завантажились на сервер. Для цього зайдемо на сторінку проекту на сайті GitLab:


Олег Іларіонов > tcp-**lab08** > Подробная информация

**tcp-**lab08**** 
Project ID: 11906719


  Star 0  Fork 0  Clone

 [Добавить лицензию](#)  1 Commit  1 Branch  0 Tags  0 **байта** Files


master 



tcp-**lab08** / 


История


 Найти файл


Web IDE





 **First commit**
Олег Іларіонов создан 21 часов назад 875f995d 


 Add README


 Add CHANGELOG





 Add CONTRIBUTING

 Включить Auto DevOps

 Добавить Kubernetes кластер

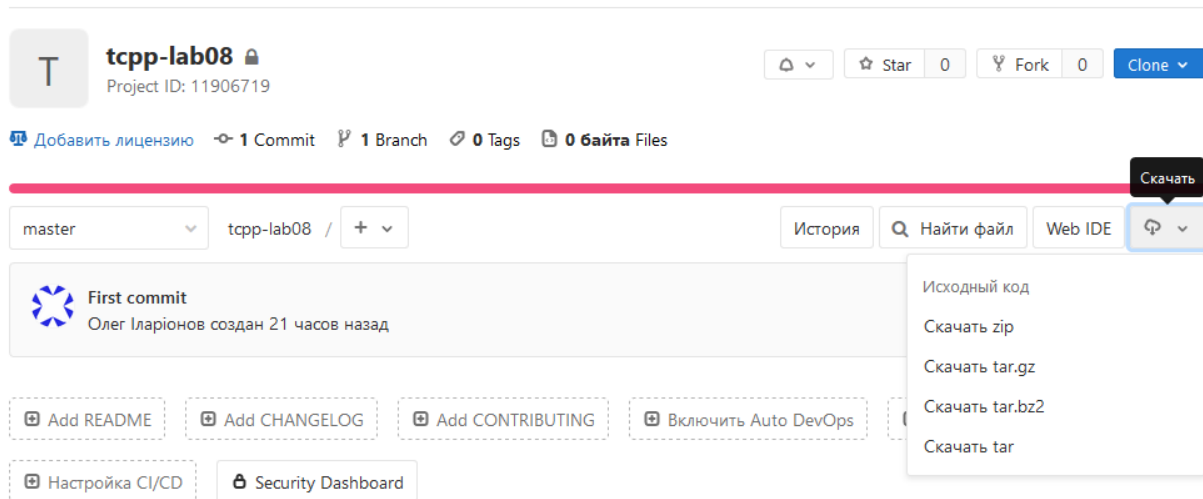
 Настройка CI/CD

 Security Dashboard

Наименование	Последний коммит	Последнее обновление
 Debug	First commit	21 часов назад
 TCP- lab8	First commit	21 часов назад
 x64/Debug	First commit	21 часов назад
 TCP- lab8.sln	First commit	21 часов назад

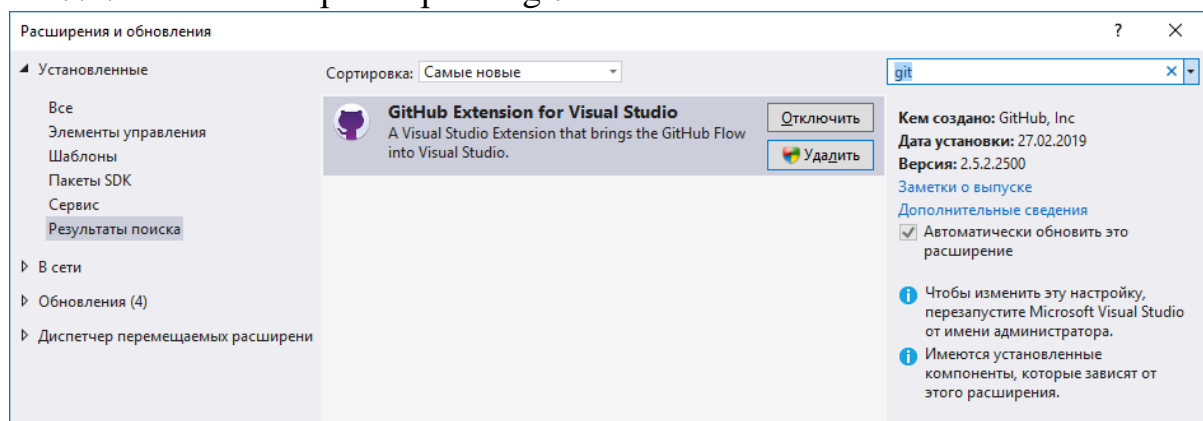
Натиснувши на посилання «Files» побачимо усі файли, які було завантажено на сервер.

Для того, щоб скачати у вигляді архіву файли репозиторію потрібно натиснути на кнопку «Download»:



7. Додавання розширення Git до Visual Studio 2017

7.1. Встановити розширення git



7.2. Виконати аналогічні дії для п.5.2 створюючи коміти за допомогою розширення git tools for Visual Studio 2017 [3]

Список літератури

1. Scott C. Pro Git [Електронний ресурс] // Режим доступу: <https://git-scm.com/book/ru/v2>
2. Введение в систему контроля версий Git [Електронний ресурс] // Режим доступу: <https://3ddd.ru/users/kokoasfalt>
3. Андреев Д. Разработка, подключение и публикация с помощью Visual Studio и Git [Електронний ресурс] // Режим доступу: [https://docs.microsoft.com/ru-ru/previous-versions/dn275834\(v=msdn.10\)](https://docs.microsoft.com/ru-ru/previous-versions/dn275834(v=msdn.10))