Лабораторна робота № __

Тема: Використання Git та GitHub у проєктах C#

🎯 Мета роботи

Навчитися використовувати систему контролю версій **Git** під час розробки **C#-проєктів**, а саме:

- створювати Git-репозиторій для C#-проєкту;
- фіксувати зміни у коді;
- переглядати історію комітів;
- відновлювати файл Program.cs з попереднього коміту;
- публікувати C#-проєкт на GitHub.

🛠️ Обладнання та програмне забезпечення

- ПК з Windows
- **.NET SDK**
- **Visual Studio** або **VS Code**
- **Git**
- Обліковий запис **GitHub**

📂 Завдання до виконання

🔷 Завдання 1. Створення C#-проєкту

1. Створити папку LabGitCSharp.
2. У терміналі виконати:
3. dotnet new console
4. Відкрити проєкт у Visual Studio або VS Code.
5. Запустити програму та переконатися, що вона працює.

🔷 Завдання 2. Ініціалізація Git

1. У папці проєкту виконати:
2. git init
3. Перевірити стан:
4. git status

🔷 Завдання 3. Перший коміт

1. Відкрити файл Program.cs.
2. Змінити код, наприклад:

Console.WriteLine("Hello, Git!");

3. Додати файли до staging:
4. git add .
5. Зробити коміт:
6. git commit -m "Перший коміт: базовий C# проєкт"

🔷 Завдання 4. Другий коміт

1. Додати введення з клавіатури:
Console.Write("Введіть ім'я: ");
string name = Console.ReadLine();
Console.WriteLine($"Привіт, {name}!");
2. Зберегти зміни.
3. Створити новий коміт:
4. git add Program.cs
5. git commit -m "Додано введення імені"

🔷 Завдання 5. Перегляд історії комітів

1. Переглянути список комітів:
2. git log --oneline
3. Записати хеш першого коміту у звіт.

🔷 Завдання 6. Відновлення файлу з попереднього коміту

1. Повернути файл Program.cs до першого коміту:
2. git restore --source <commit_id> Program.cs
3. Запустити програму та перевірити результат.
4. Зберегти відновлення новим комітом:
5. git add Program.cs
6. git commit -m "Відновлено Program.cs з попереднього коміту"

🔷 Завдання 7. Робота з GitHub

1. Створити репозиторій на GitHub.
2. Підключити його:
3. git remote add origin https://github.com/username/LabGitCSharp.git
4. Відправити код:
5. git push -u origin main

📋 Звіт повинен містити

- тему та мету лабораторної роботи;
- скріншоти виконання команд Git;
- вміст файлу Program.cs;
- історію комітів (git log --oneline);
- посилання на GitHub-репозиторій;
- висновки.

📌 Контрольні запитання

1. Що таке Git і для чого він потрібен у C#-проєктах?
2. Що таке коміт?
3. Для чого використовується файл .gitignore у .NET-проєктах?
4. Як повернути Program.cs до попередньої версії?
5. Чим відрізняється локальний репозиторій від GitHub?

Додаткове завдання
Робота з гілками Git у C#-проєкті

✅ **Крок 1. Створити нову гілку feature**
У папці C#-проєкту виконай:
git checkout -b feature
Перевір поточну гілку:
git branch
👉 Активна гілка буде позначена * feature

✅ **Крок 2. Додати метод для обчислення квадрата числа**
Відкрий файл **Program.cs** та додай метод:

```
static int Square(int x)
{
    return x * x;
}
```

Повний приклад Program.cs:

```
using System;

class Program
{
    static int Square(int x)
    {
        return x * x;
    }

    static void Main()
    {
        Console.Write("Введіть число: ");
        int n = int.Parse(Console.ReadLine());

        Console.WriteLine($"Квадрат числа: {Square(n)}");
    }
}
```

✅ **Крок 3. Зберегти зміни у гілці feature**
git add Program.cs
git commit -m "Додано метод Square()"

✅ **Крок 4. Перейти на гілку main**
git checkout main

✅ **Крок 5. Об’єднати гілку feature з main**
git merge feature

Якщо конфліктів немає — злиття відбудеться автоматично.

✅ Крок 6. Перевірка

1. Запусти програму:
2. dotnet run
3. Переконайся, що метод Square() працює.

📌 Що має бути у звіті

- команди створення гілки feature;
- код методу Square(int x);
- коміт у гілці feature;
- команда git merge feature;
- скріншот успішного злиття;
- короткий висновок.

🧠 Коротко

git checkout -b feature
git add Program.cs
git commit -m "Додано метод Square()"
git checkout main
git merge feature

Ось **чітке покрокове виконання додаткового завдання (C# + Git)** — можна прямо вставляти в лабораторну як **Завдання 8** 👇

🔷 Завдання 8. Робота з гілками (Branching)

🎯 Мета

Навчитися створювати гілки, виконувати розробку в окремій гілці та об’єднувати її з основною гілкою main.

1️⃣ Створення нової гілки

У терміналі виконати:
git checkout -b feature
Перевірити активну гілку:
git branch
Активна гілка буде позначена * feature.

2️⃣ Додавання методу для обчислення квадрата числа

Відкрити файл Program.cs і додати метод:

```
static int Square(int x)
{
    return x * x;
}
```

Оновлений приклад Program.cs:

```
using System;

class Program
{
    static int Square(int x)
    {
        return x * x;
    }

    static void Main()
    {
        Console.Write("Введіть число: ");
        int n = int.Parse(Console.ReadLine());

        Console.WriteLine($"Квадрат числа: {Square(n)}");
    }
}
```

3️⃣ Коміт змін у гілці feature

git add Program.cs
git commit -m "Додано метод Square для обчислення квадрата"

4️⃣ Об’єднання гілки feature з main

Перейти на гілку main:
git checkout main
Об’єднати гілку:
git merge feature
Якщо конфліктів немає — Git автоматично виконає злиття.

5️⃣ Перевірка результату

1. Запустити програму.
2. Переконатися, що метод Square() працює.
3. Перевірити історію:

git log --oneline --graph

✅ **Очікуваний результат**

- Створена гілка feature
- Метод Square(int x) доданий у C#-проєкт
- Гілка feature успішно об’єднана з main
- Історія комітів збережена