

# git 협업과 기여



About..

컴퓨터소프트웨어공학과  
김 원 일



# 협업과 기여 방식



## • 공개 소프트웨어 개발

- github는 공개 소프트웨어를 다수의 개발자가 개발하여 향상시키는 것이 목표
- github/gitlab과 같은 서버에 저장소를 공유하며 협업 또는 기여

## • 기여

- 단발적인 기여 또는 지속적인 기여가 가능
- 기여를 지속하면 해당 프로젝트 기여자로 등록하여 협업형태로 발전 가능
- 기여 내용은 저장소의 소유자에 의해 적용 여부가 결정
  - 적용 되면 소스 코드에 적용되고, 거부되면 수정 요구나 맞지 않는다고 판단될 수 있음

## • 협업

- 저장소에 접근할 수 있는 개발자/협업자에게 저장소 접근 권한을 부여
- 권한을 부여 받은 협업자는 저장소를 자신의 저장소처럼 사용하며 협업
- 다른 개발자들과 저장소를 공유하므로 저장소 변경에 주의가 필요함

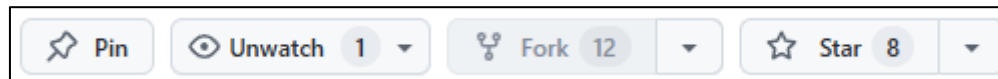


# 소프트웨어 개발 기여



## • github 활용

- github가 가장 대표적으로 소프트웨어에 기여할 수 있는 플랫폼
- 관심 있는 저장소가 있다면 Pin, Watch 등으로 변화 확인 가능
- 소스 코드 분석 등으로 개선될 수 있거나 버그로 보이는 코드를 확인
  - 개선이나 버그 수정이 필요한 경우에 저장소 소유자에게 해당 정보 전달 가능
  - "Issues" 메뉴를 이용하여 관련된 정보를 제공하여 개선을 도울 수 있음
- 본인이 소스 코드를 직접 수정하여 성능 개선 등을 수행할 수도 있음
  - 소스 코드를 본인의 저장소로 복제하고, 수정하여 소유자에게 수정 코드 전달 가능





# Issue 메뉴 활용 - 1

## • 개선 정보 전달

- 소프트웨어에 대한 수정 사항이나 개선 사항에 대해 게시판 형태로 전달
- 소유자에게 소스에 대한 다양한 의견이나 질문 등을 수행할 수도 있음
- Open은 진행되고 있는 의견을 Close는 처리되었거나 적용/거부된 의견


The screenshot shows the GitHub Issues page for the repository `markany-linux/openCode?`. The top navigation bar includes links for Code, Issues (highlighted with a red box and showing 5 issues), Pull requests, Actions, Projects, Wiki, Security, and Insights. Below the navigation bar, there is a message encouraging contribution and a link to the contributing guidelines. A search bar contains the filter `is:issue state:open`. To the right of the search bar are buttons for Labels, Milestones, and a green 'New issue' button (highlighted with a red box). Below the search bar, there are tabs for 'Open' (5) and 'Closed' (1). A table of issues is displayed with columns for Author, Labels, Projects, Milestones, Assignees, and a sort dropdown set to 'Newest'. The issues listed are:

Issue Title	Author	Opened On	Comments
[Build Server]	markany-linux	Dec 11, 2020	27
사용자 정의 타입 <code>mild_xxx</code> 에 관하여	qoor	Jun 26, 2020	1
<code>lib_utility/string_parser.c</code> 길이 값 체크 관련( <code>len_</code> 파라미터 관련 조건문)	qoor	Jun 26, 2020	2
<code>lib_utility/string_parser.c</code> 길이 체크 누락	qoor	Jun 26, 2020	1
<code>lib_utility</code> 의 인터페이스 함수들에 관하여	qoor	Jun 26, 2020	1



## • 의견 게시

- 게시판 형태로 의견 제공이 되며, 간단하게 작성도 가능

 Create new issue


Add a title \*

Add a description

WritePreview

H B I | [List Icons] | [Link Icon] | [More Icons] | [Mentions Icon] [Share Icon] [Undo Icon] [Redo Icon]

Type your description here...

 Paste, drop, or click to add files

☐ Create more



## • 의견 교환

- 게시판 형태이므로, 의견에 대해 추가적인 토론도 가능하도록 구성되어 있음
- 저장소 소유자는 의견 반영이나 거부 등의 의사에 따라 해당 이슈 처리를 결정
- 소유자는 결정에 따라 해당 이슈를 "Close"로 변경할 수 있음
  - 이슈가 "Close" 상태가 되면 더 이상 의견 게시를 할 수 없도록 버튼 들이 비활성화 됨

### 사용자 정의 타입 mild\_xxx에 관하여 #4

Open

qoor opened on Jun 26, 2020

...

mild\_null은 NULL 키워드, mild\_u32 등의 정수형 타입들은 이미 stdint.h 표준 헤더에 uint32\_t 등이 있습니다.

Open Source Project인만큼 이미 표준에 있는 내용들은 최대한 표준을 따르는 것이 좋을 것 같습니다.

😊

unangel on Jul 10, 2020

Collaborator ...

안녕하세요. 의견 감사합니다.

현재 진행되고 있는 프로젝트에서는 커널 모듈과 응용 프로그램과 라이브러리 모두 동일한 자료형을 사용하고 있습니다. 개발자의 게으름과 편의를 위해 일단 사용하고 있는 부분인데요.

해당 부분에 대해서 정확한 정책이나 설정에 대해 결정되면 처리할 예정입니다.

예를 들어 각 빌드 단위 별로 자료형 정의 파일을 따로 선언하거나, 현재와 같이 하나의 파일에 커널/사용자에 따라 헤더 파일부터 각 자료형에 모두 매크로를 이용하는 등의 방법을 고려하고 있습니다.

본 이슈에 대해서는 추후 결정이 완료되면 close하도록 하겠으며, 의견 감사합니다.

😊



# Fork로 기여하기 - 1



## • 저장소의 Fork 선택

- 본인의 저장소로 관심 저장소를 현재 상태에서 복제하여 별도 저장소 생성
- 현재 상태를 별도의 저장소로 복제하므로 몇 가지 수정이 가능
  - 브랜치 전체 또는 주요 브랜치만 복제할 수 있는 옵션 제공
  - 복제되면 본인의 저장소 목록에 fork한 저장소가 같이 표시됨

### Create a new fork

A fork is a copy of a repository. Forking a repository allows you to freely experiment with changes without affecting the original project.

Required fields are marked with an asterisk (\*).

Owner \*

Repository name \*

ycs-wikim

straceExtractor\_fork

✔ straceExtractor\_fork is available.

By default, forks are named the same as their upstream repository. You can customize the name to distinguish it further.

Description (optional)

테스트를 위한 저장소 fork

☒ Copy the `main` branch only

Contribute back to unangel/straceExtractor by adding your own branch. [Learn more.](#)

ⓘ You are creating a fork in your personal account.

Create fork

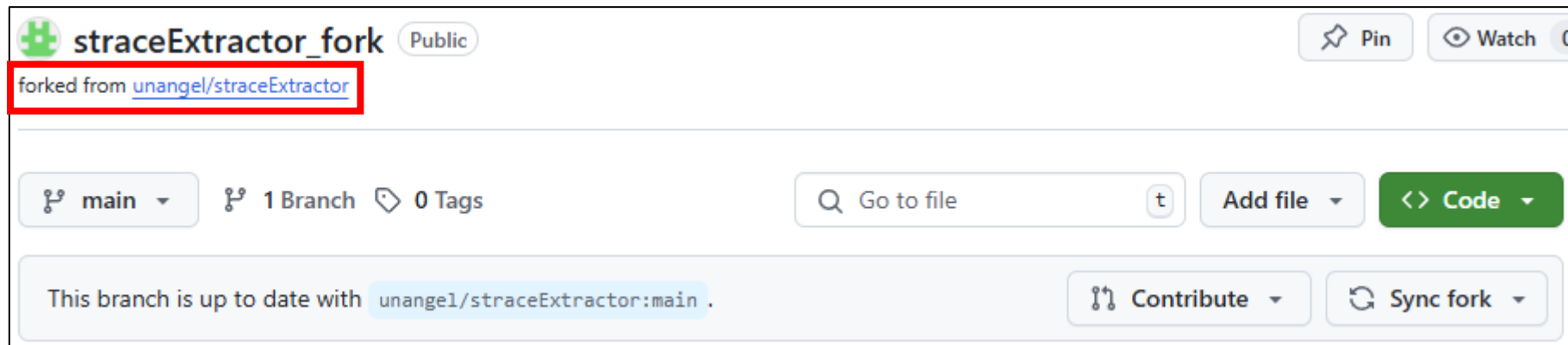


## Fork로 기여하기 - 2



### • Fork 저장소 사용

- 본인 저장소로 복제된 상태이므로 언제나 접근 가능하고 로컬로 Clone 가능
- github에서 제공하는 서비스 기능으로 기여하는 방법을 제공
- Fork를 통해 개인 저장소로 복제된 저장소는 개인 저장소처럼 사용이 가능함
- 본인이 만든 저장소와 동일하게 코드 개발을 위한 추가/수정/삭제 등이 가능
- 분석 내용이나 추가 기능 등 자유롭게 저장소 코드를 사용
- 복제된 상태이므로 본인이 자유롭게 사용 가능하나 원본 위치는 항상 표시
- 추가 기능 개발 또는 버그 수정 등을 한 경우 원작자에게 이를 알릴 수 있음







# Fork로 기여하기 - 3



## • 저장소 정보

- Fork 수행한 원본 저장소 정보를 메인 화면에서 표시
- 원본 저장소와 Fork된 저장소 간의 정보 비교를 자동으로 진행
- Fork된 저장소와의 정보 교환을 위한 기능들을 제공

The screenshot shows a GitHub repository page for a fork named `straceExtractor_fork`, which is public and forked from `unangel/straceExtractor`. The interface includes a header with the repository name, a 'Public' badge, and buttons for 'Pin' and 'Watch'. Below the header, there's a section for the current branch (`main`), showing 1 branch and 0 tags. A search bar 'Go to file' and buttons for 'Add file' and 'Code' are present. A status bar indicates 'This branch is up to date with unangel/straceExtractor' and provides 'Contribute' and 'Sync fork' options. The main content area shows the initial commit by 'unangel' with files `.gitignore` and `LICENSE`. Two side-by-side comparison boxes are shown: the left one indicates 'This branch is not ahead of the upstream' (unangel/straceExtractor:main) with a minus sign icon, and the right one indicates 'This branch is not behind the upstream' (unangel/straceExtractor:main) with a checkmark icon. Both boxes state 'No new commits yet. Enjoy your day!'.



## Fork로 기여하기 - 3



### • 저장소 다운로드

- 본인 저장소로 복제되었기 때문에 본인 계정에서 다운로드
- fork를 수행한 시점의 저장소이므로 원본 저장소 업데이트 확인 필요
- 저장소 수정을 해도 원본 저장소에는 영향이 없으므로 편하게 수정 진행

```
MINGW64:/f/202507001/straceExtractor_fork
unange1@DESKTOP-I80QG85 MINGW64 /f/202507001
$ git clone https://github.com/ycs-wikim/straceExtractor_fork
Cloning into 'straceExtractor_fork'...
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 4 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (4/4), 12.73 KiB | 12.73 MiB/s, done.

unange1@DESKTOP-I80QG85 MINGW64 /f/202507001
$ cd straceExtractor_fork/

unange1@DESKTOP-I80QG85 MINGW64 /f/202507001/straceExtractor_fork (main)
$ ls
LICENSE

unange1@DESKTOP-I80QG85 MINGW64 /f/202507001/straceExtractor_fork (main)
$
```



## Fork로 기여하기 - 4



- 소스 수정 후 업로드

- 비어 있는 저장소 이므로 소스 파일을 추가하고 Commit 후 업로드
- 소스의 내용은 간단히 출력만 수행하는 코드
- 내용 수정 후의 상태 확인을 위한 실습이므로 굳이 코드로 작성하지 않아도 됨

```
MINGW64:/f/202507001/straceExtractor_fork

[main c789c10] 1. main.c [ + ] add main.c file for main function
1 file changed, 7 insertions(+)
create mode 100644 main.c

unangel@DESKTOP-I80QG85 MINGW64 /f/202507001/straceExtractor_fork (main)
$ git push
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 16 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 397 bytes | 397.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/yys-wikim/straceExtractor_fork
25044e3..c789c10 main -> main

unangel@DESKTOP-I80QG85 MINGW64 /f/202507001/straceExtractor_fork (main)
$
```



# Fork로 기여하기 - 5

## • 업로드한 Fork 저장소

- 저장소 내용 변경을 원본 저장소에 기여할 수 있는 메뉴의 활성화
- "Open pull request"로 원본 저장소에 기여 가능함을 표시

The screenshot shows a GitHub repository page for 'straceExtractor\_fork', which is a public fork of 'unangel/straceExtractor'. The repository is currently on the 'main' branch, which is 1 commit ahead of the upstream 'unangel/straceExtractor:main'. A tooltip is displayed over the 'Contribute' button, indicating that the branch is 1 commit ahead and suggesting to 'Open a pull request to contribute your changes upstream.' The 'Open pull request' button in the tooltip is highlighted with a red rectangle.

straceExtractor\_fork Public

forked from [unangel/straceExtractor](#)

main 1 Branch 0 Tags

Go to file Add file Code

This branch is 1 commit ahead of unangel/straceExtractor:main . Contribute Sync fork

ycs-wikim 1. main.c

.gitignore	Initial commit
LICENSE	Initial commit
main.c	1. main.c

README License

This branch is 1 commit ahead of unangel/straceExtractor:main .


Open a pull request to contribute your changes upstream.

Open pull request

- **저장소 정보와 일반적인 게시판 작성과 같이 내용 전달**

## Open a pull request

Create a new pull request by comparing changes across two branches. If you need to, you can also [compare across forks](#). [Learn more about diff](#)



base repository: **unangel/straceExtractor** ▼


base: **main** ▼

← ...

head repository: **ycs-wikim/straceExtractor\_fork** ▼

compare: **main** ▼

✓ **Able to merge.** These branches can be automatically merged.



### Add a title

1. main.c



### Add a description

Write Preview

H B I ≡ <> 🔗 | ☰ ☷ ≡ | 📎 @ ↻ ↺

[ + ] add main.c file for main function

Commit 메시지의 내용이 기본으로 설정

 Markdown is supported  Paste, drop, or click to add files

☒ Allow edits by maintainers ?

Create pull request ▼



## • 요청 정보

- Pull request는 원본 저장소에 등록되어 기여를 직접적으로 수행
  - 원작자의 확인을 통해 코드를 원본 저장소에 등록할지 여부를 결정
  - 확인을 위한 여러 가지 정보를 원작자에게 제공

Filters is:pr is:open Labels 9 Milestones 0 New pull request

1 Open 0 Closed Author Label Projects Milestones Reviews Assignee Sort

1. main.c 원작자에게는 목록으로 나타남  
#1 opened 6 minutes ago by ycs-wikim

unangel / straceExtractor 세부 내용에서 정보 획득과 정보 교환 가능

<> Code Issues Pull requests 1 Actions Projects Security Insights

1. main.c #1

Open ycs-wikim wants to merge 1 commit into unangel:main from ycs-wikim:main

Conversation 0 Commits 1 Checks 0 Files changed 1

ycs-wikim commented now

[ + ] add main.c file for main function

1. main.c c789c10



# Fork로 기여하기 - 8

## • 원작자 페이지

- 병합을 수행할 것인지 별도 Comment를 입력할 것인지 결정 가능
- 병합 시 어떠한 형태로 병합을 수행할 것인지를 선택할 수 있음

Conversation 0 Commits 1 Checks 0 Files changed 1

ycs-wikim commented 9 minutes ago First-time contributor ...

[ + ] add main.c file for main function

1. main.c c789c10

✓ No conflicts with base branch  
Merging can be performed automatically.

Merge pull request You can also merge this with the command line.  
[View command line instructions.](#)

✓ Create a merge commit  
All commits from this branch will be added to the base branch via a merge commit.

Squash and merge  
The 1 commit from this branch will be added to the base branch.

Rebase and merge  
The 1 commit from this branch will be rebased and added to the base branch.

Add a comment

Write Preview

Add your comment here

Markdown is supported Paste, drop, or click to add files

Close pull request Comment

Commit message

Merge pull request #1 from ycs-wikim/main

Extended description

1. main.c

This commit will be authored by unangel@users.noreply.github.com.

Confirm merge Cancel

Pull request successfully merged and closed

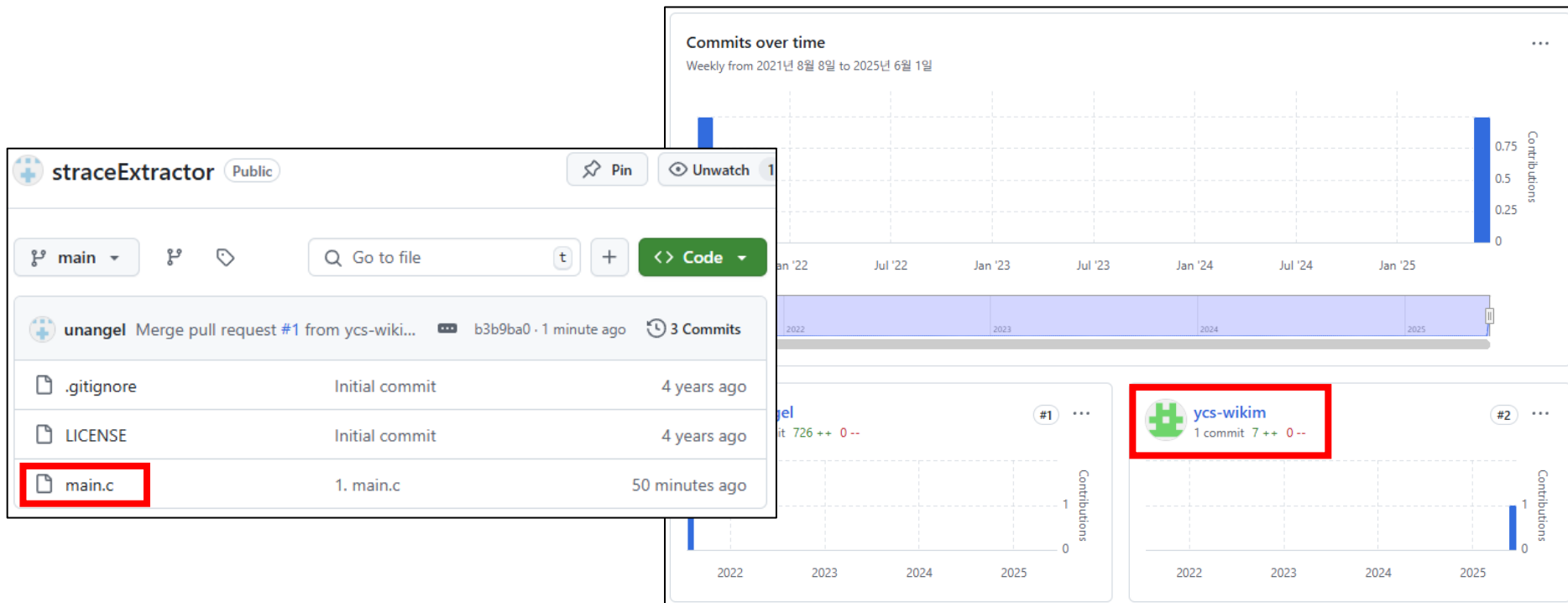
You're all set — the branch has been merged.



# Fork로 기여하기 - 9

## • 기여 확인

- 소스 코드에 Commit 한 내용이 적용되는 것을 확인할 수 있음
- “Insights” 메뉴의 “Contributors” 항목에 기여자로 추가된 것을 확인
- 이와 같은 형태로 권한 없는 저장소에 Fork를 이용하여 기여할 수 있음





# git 협업 - 1



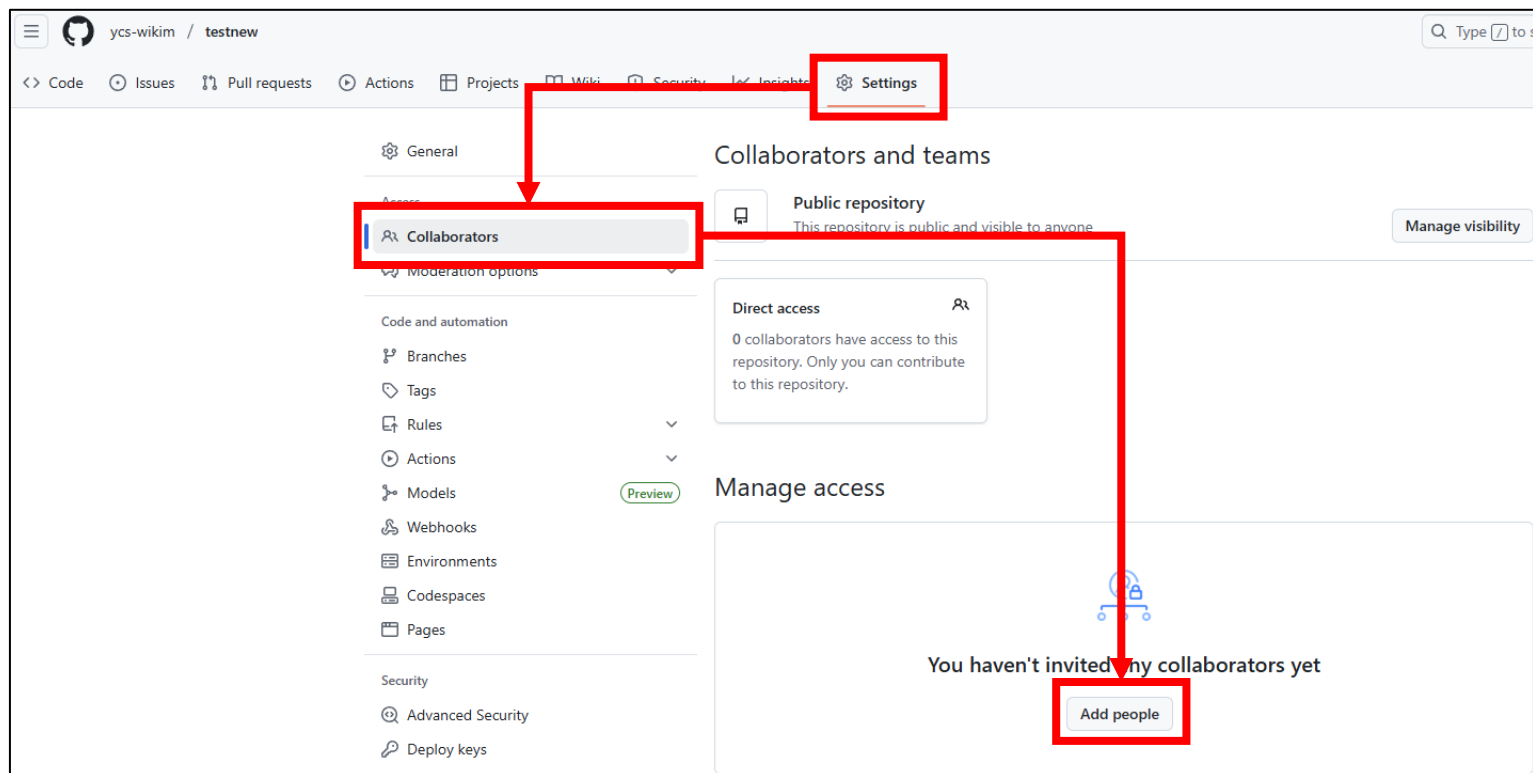
## • 개발 협업

- 개발 목표가 동일한 사람들이 협동하여 개발을 진행하는 것
- 설계를 먼저 진행하여 개발에 필요한 내용들을 미리 정리
- 소프트웨어 개발 설계에서 버전 관리 시스템을 적용하는 방법
  - 개발에 필요한 기능들을 정리 : 기능 요구사항과 비 기능 요구사항으로 정리
  - 요구사항에 따라 개발할 클래스 또는 기능 별로 파일들을 구성
  - 각자 맡은 기능에 대해 비 선형적 개발을 진행
  - 개발 상황 공유와 기능 개발에 관련된 문제점과 해결 방법 등을 협동하여 해결
  - 개발된 소스 코드를 하나로 병합하고, 발생하는 문제점들을 해결
  - 프로그램의 동작 확인을 위한 테스트와 버그 등을 수행
- 기능 요구사항 : 사용자가 요청하거나 구현을 위해 필요한 기능 (파일 저장 기능)
- 비 기능 요구사항 : 기능 요구사항을 개발하기 위해 필요한 기능 (저장 및 확인 기능)
- 하나의 저장소를 활용하여 개발을 진행
  - 저장소 하나를 공유하여 사용하기 때문에 다양한 문제가 발생할 수 있으므로 주의
  - 주요 브랜치인 main 또는 master 브랜치는 팀장이나 책임자 또는 숙련자가 수행
  - 개발자는 반드시 브랜치를 생성하여 개발하고, 팀장이 병합

## git 협업 - 2

### • github를 이용한 협업

- 기여 방식을 이용한 협업도 가능하지만 코드 병합 방식이 불편
- 하나의 저장소를 여러 개발자가 공유하여 협업하는 방식
- 팀장 또는 책임자가 저장소를 생성하고, 협업자에게 권한을 할당하는 방식
- 저장소 설정에 협력자(Collaborators)를 등록하는 형태



## git 협업 - 3


### • 협력자 추가 - 1


- 협력자로 등록할 사용자를 검색하여 등록 가능
- 사용자 아이디 또는 사용자의 전자 메일 주소를 통해 등록 가능
- 아이디를 알고 있는 경우에는 목록에서 선택하여 등록을 요청
- 전자 메일을 메일을 보내서 등록을 요청하는 형태


Add people to testnew


Search by username, full name, or email

Q ycs

 ycs  
Invite collaborator

 ycs  
ycs0403 • Invite collaborator

 ycs-201807006  
ycs-201807006 • Invite collaborator

 YCS\_Forever  
YCS-Forever • Invite collaborator


Cancel

Add to repository

Add people to testnew

Search by username, full name, or email

Q unangel@yuhan.ac.kr

 unangel@yuhan.ac.kr  
Invite to testnew

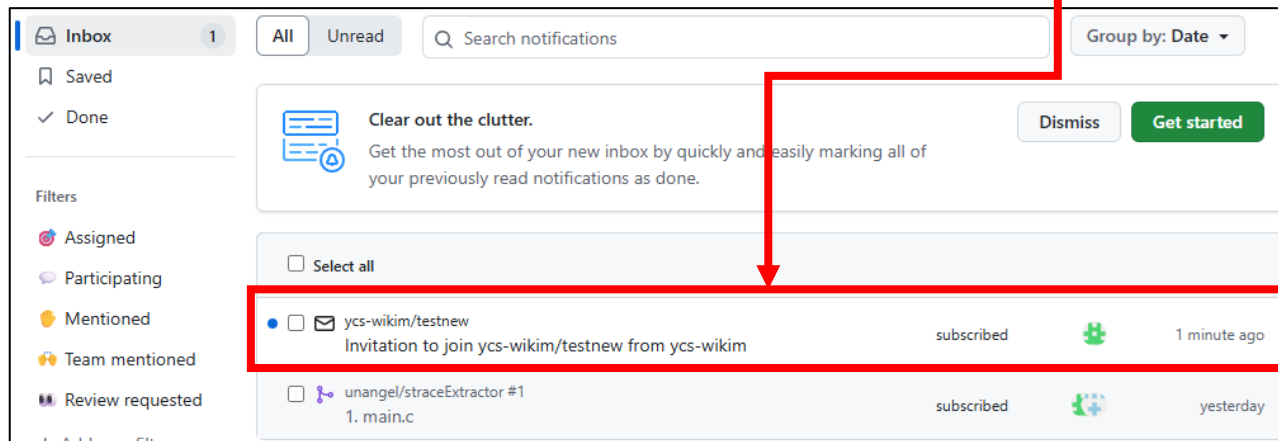
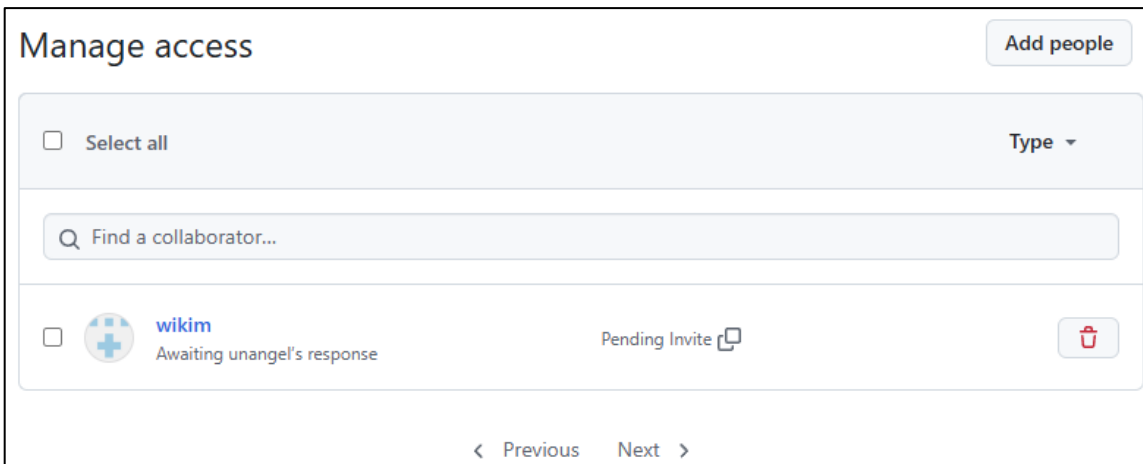
Cancel

Add to repository

## git 협업 - 4

### • 협력자 추가 - 2

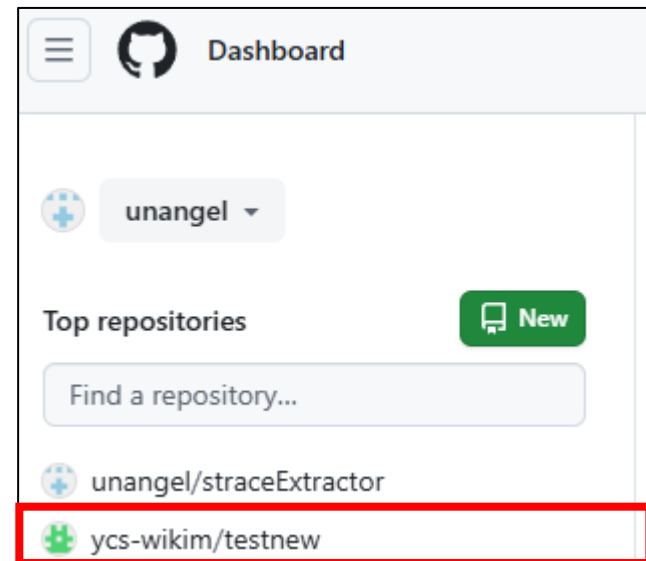
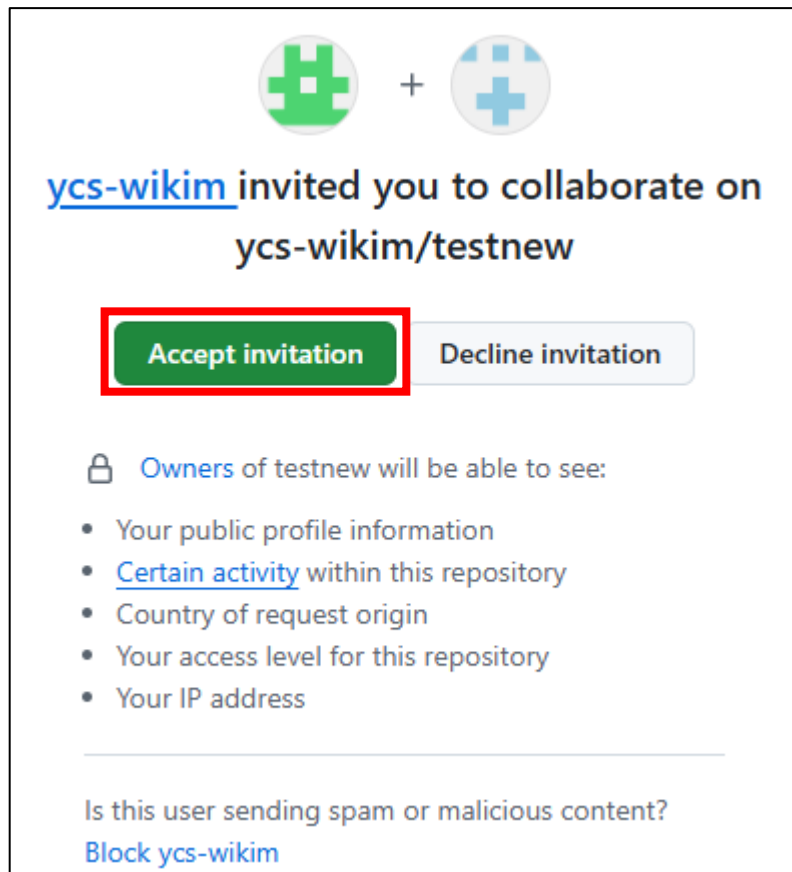
- 협력자로 추가를 요청하면 협력자의 승인을 대기하며, 취소도 가능
- 요청을 받은 사용자는 github로 전달된 요청 정보를 확인



## git 협업 - 5

### • 협력자 추가 - 3

- 협력자 요청에 대해 승인 또는 거부할 수 있음
- 승인할 경우, 해당 저장소에 접근 권한이 생기고, 본인 저장소에도 나타남



## git 협업 - 6

### • 협력자 추가 - 4

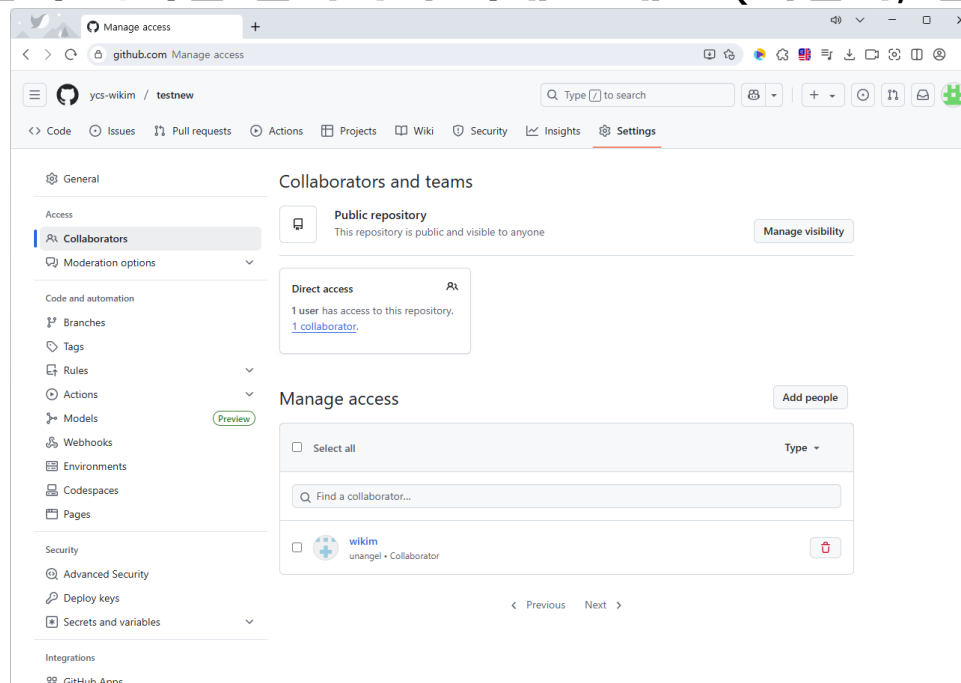
- 저장소 소유주의 협력자 메뉴에 협력자로 등록된 것을 확인 가능
- 저장소는 소유주의 것이므로 협력자를 삭제하는 것도 가능
- 다수 개발자를 이러한 방식으로 협력자로 초대할 수 있으며 협업 가능





## • 실습 과제

- 1명 이상의 팀원을 모아서 협력자로 저장소에 등록해 볼 것
- 실습을 진행할 팀원들을 먼저 모을 것
  - 지속적으로 유지할 팀이 아니므로 편하게 옆자리 학생과 진행해도 문제 없음
- 저장소를 생성할 책임자 선택 : 저장소 이름은 "testCollaborators"로 고정
- 저장소 생성 후, 교재 설명과 같이 협력자로 팀원들을 등록
- 등록된 학생들의 목록을 캡처하여 과제로 제출 (책임자, 팀원 동일한 파일로)



# git 프로젝트 실습



About..

컴퓨터소프트웨어공학과  
김 원 일





# 개인 프로젝트 - 계산기



## • 개인 프로젝트 실습

- 사칙 연산과 나머지 연산을 각각 함수로 구현하는 프로젝트
- +, - 함수는 인수를 전달받은 순서를 그대로 수행
- \*, /, % 함수는 +, - 함수를 이용하여 구현
  - Ex)  $3 * 5 == 3 + 3 + 3 + 3 + 3$
- 각 함수는 함수 이름과 동일한 파일을 생성하고, 각자 구현
- 파일명에 해당하는 브랜치를 생성하고, 병합하여 프로그램 실행을 확인
- 연산에 대응하는 파일명과 브랜치 이름
  - main : calc.cpp - main or master 브랜치
  - + : add.h/cpp - dev/basic 브랜치
  - - : sub.h/cpp - dev/basic 브랜치
  - \* : mul.h/cpp - dev/mul 브랜치
  - / : dive.h/cpp - dev/dive 브랜치
  - % : mod.h/cpp - dev/mod 브랜치



## • 원격 저장소 생성

- github 개인 저장소에 아래와 같이 개인 저장소를 생성하고 로컬로 복제
- 저장소

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository [Import a repository](#).

Required fields are marked with an asterisk (\*).

Owner \*  /    
 ✓ calc is available.

Great repository names are short and memorable. Need inspiration? How about [laugh](#)

Description (optional)

☒ Public  
Anyone on the internet can see this repository. You choose who can commit.

☐ Private  
You choose who can see and commit to this repository.

Initialize this repository with:

☒ Add a README file  
This is where you can write a long description for your project. [Learn more about READMEs](#).

Add .gitignore

Choose which files not to track from a list of templates. [Learn more about ignoring files](#).

Choose a license

License:

```
MINGW64:/f/202507001/calc
unangel@DESKTOP-I8OQG85 MINGW64 /f/202507001
$ git clone https://github.com/ycs-wikim/calc
Cloning into 'calc'...
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 (from 0)
Receiving objects: 100% (5/5), 15.75 KiB | 3.15 MiB/s, done.

unangel@DESKTOP-I8OQG85 MINGW64 /f/202507001
$ cd calc/

unangel@DESKTOP-I8OQG85 MINGW64 /f/202507001/calc (main)
$ |
```



## • Visual Studio 프로젝트 생성

- 저장소를 복제한 위치의 상위 디렉터리를 설정하고, 프로젝트 이름 설정
- 저장소 자체에 프로젝트를 설정하는 방법으로 관리가 자동으로 시작
- Visual Studio 파일 제외까지 적용되어 소스 코드 파일만 관리되는 상태

### 새 프로젝트 구성

콘솔 앱 C++ Windows

프로젝트 이름(J)  
**calc**

위치(L)  
**F:\202507001**

솔루션 이름(M) ⓘ  
calc

☐ 솔루션 및 프로젝트를 같은 디렉터리에 생성

"F:\202507001\calc\calc"에 프로젝트를(가) 만들어줍니다.

```
MINGW64:/f/202507001/calc
unangel@DESKTOP-I8OQG85 MINGW64 /f/202507001/calc (main)
$ ls
LICENSE  README.md  calc/  calc.sln

unangel@DESKTOP-I8OQG85 MINGW64 /f/202507001/calc (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

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

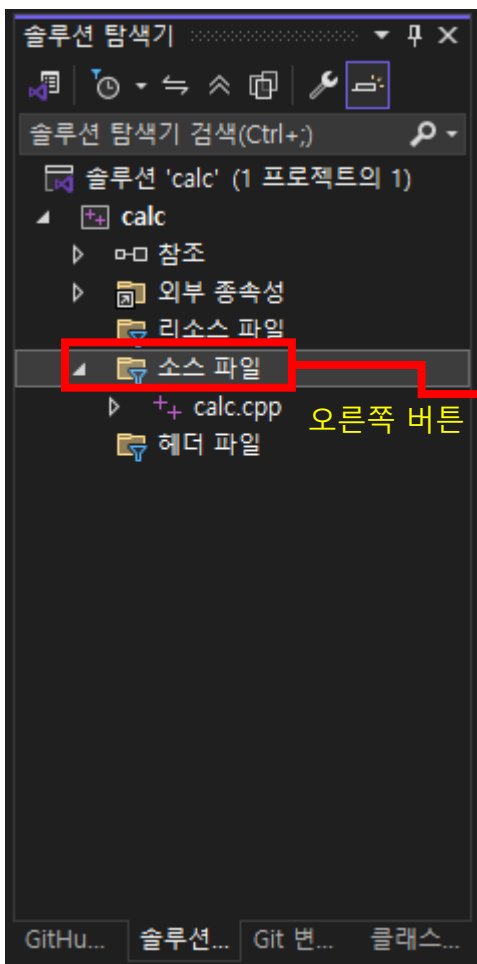
nothing added to commit but untracked files present (use "git add" to track)

unangel@DESKTOP-I8OQG85 MINGW64 /f/202507001/calc (main)
$
```

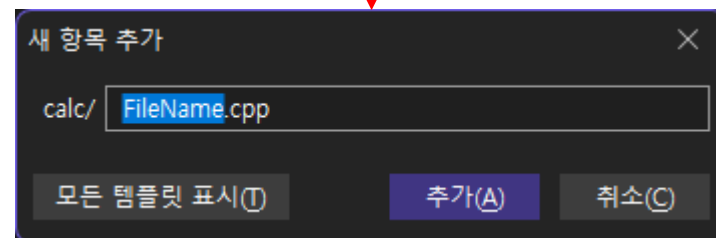
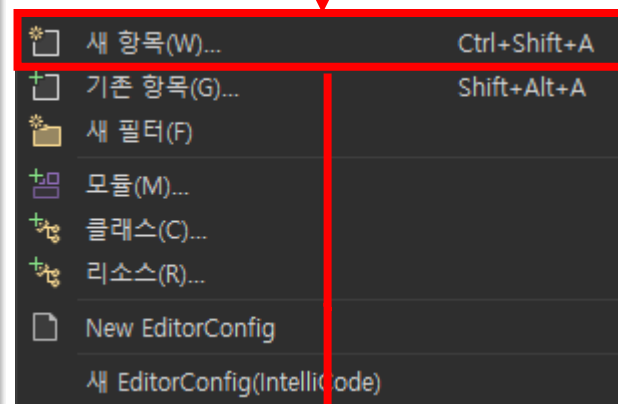
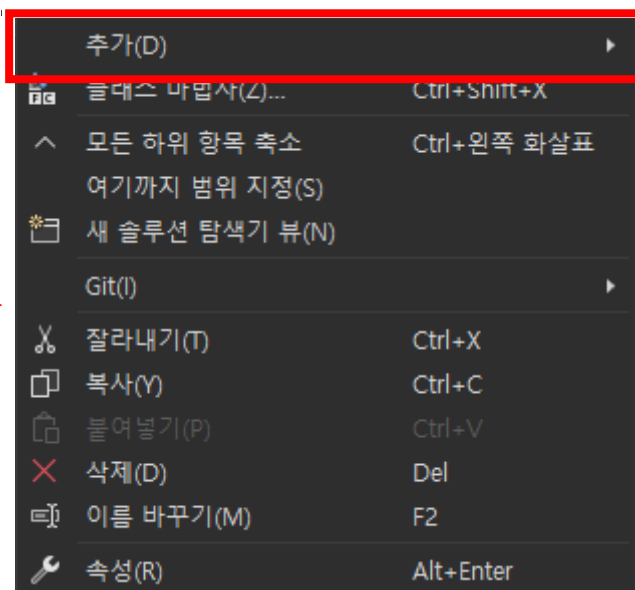


## • 프로젝트 파일 추가

- 설계에서 제시한 파일들을 모두 생성. 소스 파일과 헤더 파일에 모두 추가



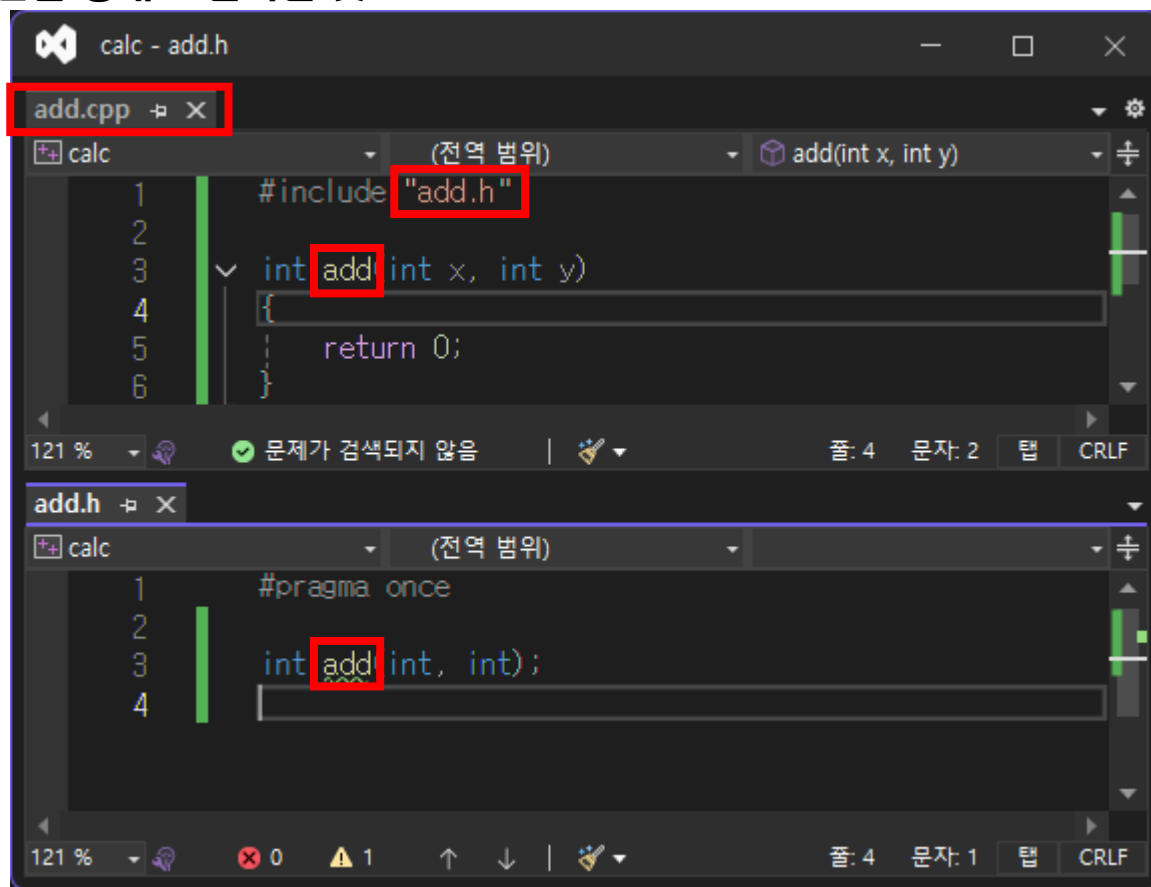
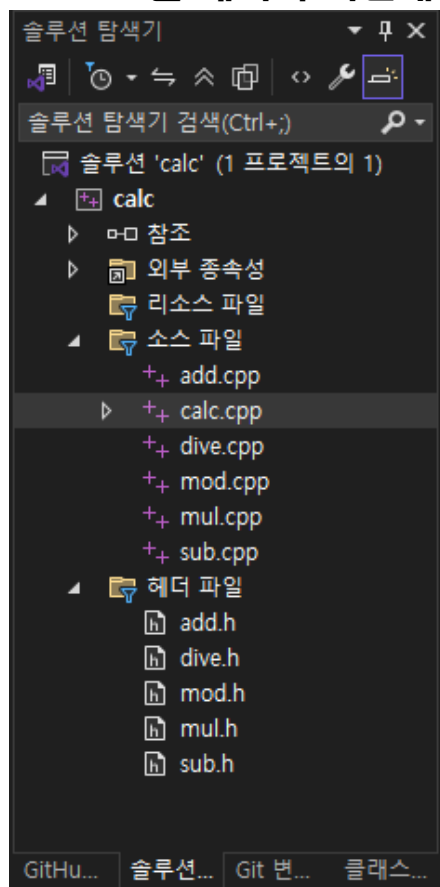
오른쪽 버튼





## • 헤더/구현 파일에 내용 추가하기

- 모든 헤더 파일(.h)에는 원형을, 구현 파일(.cpp)에는 함수를 구현 입력
- 함수는 모두 동일한 형태로 구성하고, 반환 값도 모두 0으로 설정
  - 모든 헤더와 파일에 동일한 형태로 입력할 것





## ● 프로젝트 Commit

- 파일들을 추가한 상태에서 현재까지의 파일들을 모두 Commit 수행
- 설계에 따라 비 선형 개발을 진행하기 위한 기반 파일들을 작성한 상태
- 프로젝트 파일들을 브랜치를 생성하고, 코드 입력으로 개발 진행

```
MINGW64:/f/202507001/calc
unangel@DESKTOP-I80QG85 MINGW64 /f/202507001/calc (main)
$ git add .

unangel@DESKTOP-I80QG85 MINGW64 /f/202507001/calc (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   calc.sln
    new file:   calc/add.cpp
    new file:   calc/add.h
    new file:   calc/calc.cpp
    new file:   calc/calc.vcxproj
    new file:   calc/calc.vcxproj.filters
    new file:   calc/dive.cpp
    new file:   calc/dive.h
    new file:   calc/mod.cpp
    new file:   calc/mod.h
    new file:   calc/mul.cpp
    new file:   calc/mul.h
    new file:   calc/sub.cpp
    new file:   calc/sub.h
```

```
MINGW64:/f/202507001/calc

[main c5ed47c] 1. calc project [ + ] project base files
14 files changed, 242 insertions(+)
create mode 100644 calc.sln
create mode 100644 calc/add.cpp
create mode 100644 calc/add.h
create mode 100644 calc/calc.cpp
create mode 100644 calc/calc.vcxproj
create mode 100644 calc/calc.vcxproj.filters
create mode 100644 calc/dive.cpp
create mode 100644 calc/dive.h
create mode 100644 calc/mod.cpp
create mode 100644 calc/mod.h
create mode 100644 calc/mul.cpp
create mode 100644 calc/mul.h
create mode 100644 calc/sub.cpp
create mode 100644 calc/sub.h
```



- calc.cpp 수정 - 1

- 다른 파일들이 모두 개발될 것이라고 가정하고, 코드를 입력
- 2개의 수를 입력 받아 각 함수를 호출하도록 코드를 구성
- 코드 입력 후 실행 여부를 확인하고, main/master 브랜치에서 Commit 수행

```
✓ #include <iostream>
  #include "add.h"
  #include "dive.h"
  #include "mod.h"
  #include "mul.h"
  #include "sub.h"

✓ int main()
{
    int x = 0;
    int y = 0;

    std::cout << "첫번째 수를 입력하세요 : ";
    std::cin >> x;
    std::cout << "두번째 수를 입력하세요 : ";
    std::cin >> y;

    printf("입력된 수 x[ %d ] y[ %d ]\n", x, y);
    printf("add[ %d ] sub[ %d ] mul[ %d ] dive[ %d ] mod[ %d ]\n",
        add(x, y), sub(x, y), mul(x, y), dive(x, y), mod(x, y));
}
```



## • calc.cpp 수정 - 2

- 저장소 상태에서 수정한 calc.cpp와 프로젝트 관리 파일들의 수정을 확인
- 수정된 프로젝트 전체 상태를 저장소에 Commit 수행
- Commit된 상태에서 각 파일에 대한 코드를 브랜치를 생성하여 입력

```
MINGW64:/f/202507001/calc
unangel@DESKTOP-I80QG85 MINGW64 /f/202507001/calc (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:   calc/calc.cpp
        modified:   calc/calc.vcxproj
        modified:   calc/calc.vcxproj.filters

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

unangel@DESKTOP-I80QG85 MINGW64 /f/202507001/calc (main)
$ git add .

unangel@DESKTOP-I80QG85 MINGW64 /f/202507001/calc (main)
$ git commit
```





## • 브랜치 생성

- “dev/기능” 형태로 브랜치를 생성 4개의 기능을 각 브랜치에서 개발
  - +, -는 base에서 다른 브랜치는 해당 파일에 대한 기능을 추가
- 각 브랜치로 이동 후, 순서대로 코드를 입력하고 Commit을 수행
- 한 브랜치에서는 하나의 기능만을 구현하도록 주의할 것

```
MINGW64:/f/202507001/calculator
unangel@DESKTOP-I8OQG85 MINGW64 /f/202507001/calculator (main)
$ git branch
* main

unangel@DESKTOP-I8OQG85 MINGW64 /f/202507001/calculator (main)
$ git branch dev/base ; git branch dev/mul ; git branch dev/dive ; git branch dev/mod

unangel@DESKTOP-I8OQG85 MINGW64 /f/202507001/calculator (main)
$ git branch
dev/base
dev/dive
dev/mod
dev/mul
* main

unangel@DESKTOP-I8OQG85 MINGW64 /f/202507001/calculator (main)
$ |
```



## • 병합

- 각 브랜치의 작업이 완료되면 병합하여 코드를 통합
- 병합 순서는 관계 없으나 모든 브랜치를 main 브랜치로 병합할 것
- 병합에서 발생할 수 있는 충돌 문제도 확인해볼 것
- dev/mul, dev/mod 브랜치에서 calc.cpp을 동시에 수정한 경우에 발생
- 실행 결과가 정상적으로 동작하는지 확인

```
Microsoft Visual Studio 디버그 x + v
첫 번째 수를 입력하세요 : 8
두 번째 수를 입력하세요 : 2
입력된 수 x[ 8 ] y[ 2 ]
add[ 10 ] sub[ 6 ] mul[ 16 ] dive[ 4 ] mod[ 0 ]

F:\202507001\calc\x64\Debug\calc.exe(프로세스 9780)이(가) 0 코드(0x0)와 함께 종료되었습니다.
이 창을 닫으려면 아무 키나 누르세요...
```



## 협업 프로젝트 실습 및 과제



### • 계산기 프로젝트

- 최소 2인, 최대 4인 이하의 팀으로 구성하여 실습을 진행할 것
- 팀장: 초기 설정 후 브랜치를 나누기 전(프로젝트 설정-5)까지 동일하게 진행
- 원격 저장소에 작업한 코드를 업로드 하고, 팀원을 협력자로 등록
- 등록된 협력자 수에 따라 개발할 파일을 지정
- 팀장: main 브랜치에 프로젝트 설정-6의 코드 입력, Commit 후 push 수행
- 팀원: 팀장 push 후에 저장소를 clone하여 개발할 파일에 맞게 브랜치 생성
- 팀장/팀원: 본인이 맡은 코드를 완성하고, 서버로 브랜치를 push
- 팀장: 모든 팀원의 브랜치를 git pull로 가져온 다음 브랜치를 병합
- 병합 순서는 관계 없으며, 모두 병합된 다음 실행이 이루어지는지 확인
- 병합과 실행을 확인한 다음 병합된 내용을 push로 업로드
- 업로드 한 다음 저장소의 "Insights" → "Network"를 캡처하여 과제 제출
- 과제가 2개이므로 2개의 캡처를 파워포인트에 붙여 넣어서 제출