

C 개발 환경 구축 및 기초 문법



Visual Studio 2022



프로그래밍과 C언어

- **프로그래밍이란?**

- 컴퓨터 프로그램을 만드는 일
- 컴퓨터에게 일을 하도록 명령어를 만드는 것

- **프로그램**

- 컴퓨터에게 일을 시키는 명령의 집합 또는 프로그래밍한 작업의 결과.

- **프로그래밍 언어의 종류**

- C언어 , C++ , Java , Python, JavaScript 등

- **컴파일**

- 프로그램 언어를 컴퓨터가 알 수 있는 언어(기계어)로 바꿔 주는 일
- 컴파일러는 기계어로 번역해 주는 프로그램



컴파일러란 무엇인가?

- 컴파일(Compile)

- 컴퓨터에게 소스 코드를 이해시키기 위해 0과 1로 구성된 코드로 변환하는 과정

- 기계어

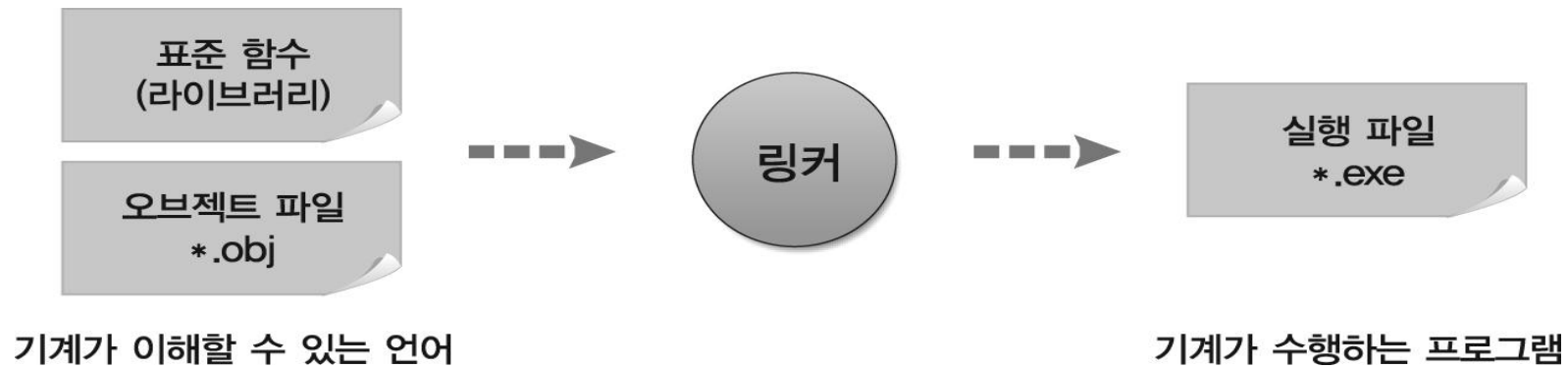
- 컴퓨터가 이해하는 **2진 숫자(0과 1)**로 작성된 언어. 이렇게 변경된 파일을 목적파일(object file)이라 한다.

- Step1



프로그램 작성 방법

– Step2



– Step3



C언어란 무엇인가?

- C언어의 탄생

- 누가 만들었나?

- 미국 AT&T사의 벨(Bell) 연구소의 데니스 리치와 켄 톰슨이 만들었다.

- 왜 만들었나?

- 유닉스(Unix) 운영체제를 개발을 위해 데니스 리치가 만든 언어였다.
 - 유닉스는 다중 사용자, 다중 작업 운영체제로 이전의 운영체제들과 달리 이식성이 뛰어났다.
 - 켄 톰슨이 처음에 B언어를 만들었고, 데니스 리치가 B언어를 업그레이드하여 C언어를 완성함.



C언어의 특징

- C언어의 특징

- C 언어는 절차지향 언어이다.

C언어의 절차지향적 특성은 작업을 단계별 함수로 분해하고 순차적으로 실행하는 방식으로, 하드웨어 제어나 시스템 프로그래밍과 같이 **명확한 흐름**이 필요한 경우에 강점을 발휘합니다. 반면, 대규모 소프트웨어 개발에는 객체지향 언어보다 유연성이 떨어질 수 있다.

- C 언어 프로그램은 이식성이 좋다.

C로 작성된 코드가 다양한 플랫폼(운영체제, 하드웨어 아키텍처 등)에서 수정 없이 또는 최소한의 변경만으로 컴파일되고 실행될 수 있다.

- C 언어는 저급언어이다.

컴퓨터 입장에서 C언어는 다른 언어들에 비해 이해하기 수월한 언어에 속한다. 이는 컴파일러에 의해 번역되고 전달되는 과정이 빠르고 간결하다는 의미이다. (사람의 입장에서 이해하기 수월한 언어를 고급언어라 한다.)



통합개발환경(IDE) – 비주얼 스튜디오

◆ 솔루션 > 프로젝트 > 파일(.c)

- 새 프로젝트 만들기 -> 빈 프로젝트 -> 프로젝트 이름

원하는 작업을 선택하세요.

최근 파일 열기(R)

시작

- 리포지토리 복제(C)
GitHub 또는 Azure DevOps 같은 온라인 리포지토리에서 코드를 가져옵니다.
- 프로젝트 또는 솔루션 열기(P)
로컬 Visual Studio 프로젝트 또는 .sln 파일을 엽니다.
- 로컬 폴더 열기(F)
폴더 내에서 코드를 탐색 및 편집합니다.
- 새 프로젝트 만들기(N)
시작하려면 코드 스캐폴딩과 함께 프로젝트 템플릿을 선택합니다.

템플릿 검색(Alt+S)(S)

모든 언어(L) 모든 플랫폼(P) 모든 프로젝트 형식(T)

빈 프로젝트
Windows용 C++를 사용하여 처음부터 시작합니다. 시작 파일을 제공하지 않습니다.
C++ Windows 콘솔

콘솔 앱
Windows 터미널에서 코드를 실행합니다. 기본적으로 "Hello World"를 출력합니다.
C++ Windows 콘솔

Windows 데스크톱 마법사
마법사를 사용하여 고유한 Windows 앱을 만드세요.
C++ Windows 데스크톱 콘솔 라이브러리

새 프로젝트 구성

빈 프로젝트 C++ Windows 콘솔

프로젝트 이름(N)
HelloC

위치(L)
C:\WC2020

솔루션 이름(M) ⓘ
HelloC

☐ 솔루션 및 프로젝트를 같은 디렉터리에 배치(D)

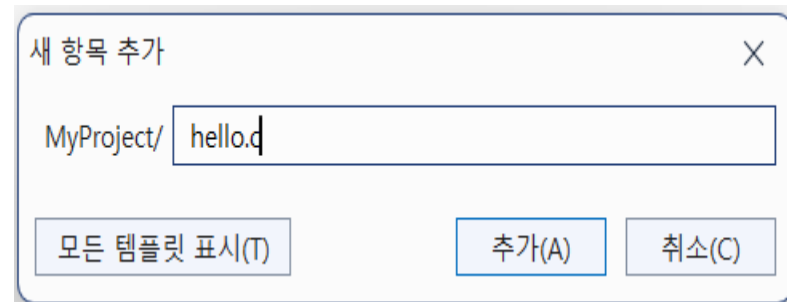
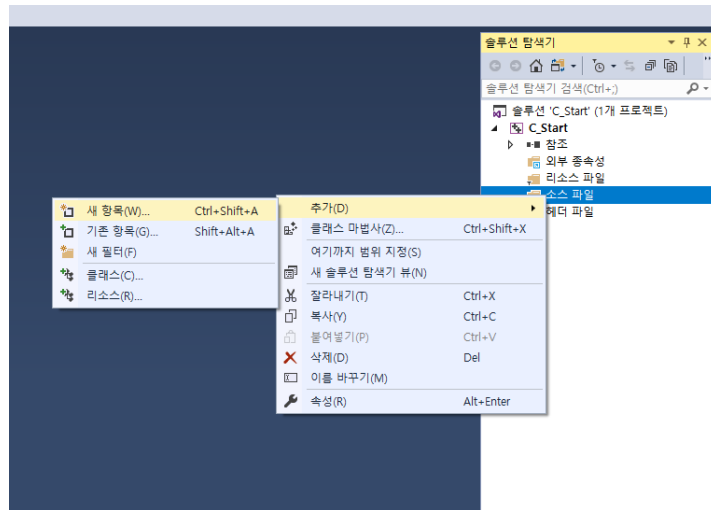
파일 저장 경로 설정
cworks 작업 디렉터리
만들기



통합개발환경(IDE) – 비주얼 스튜디오

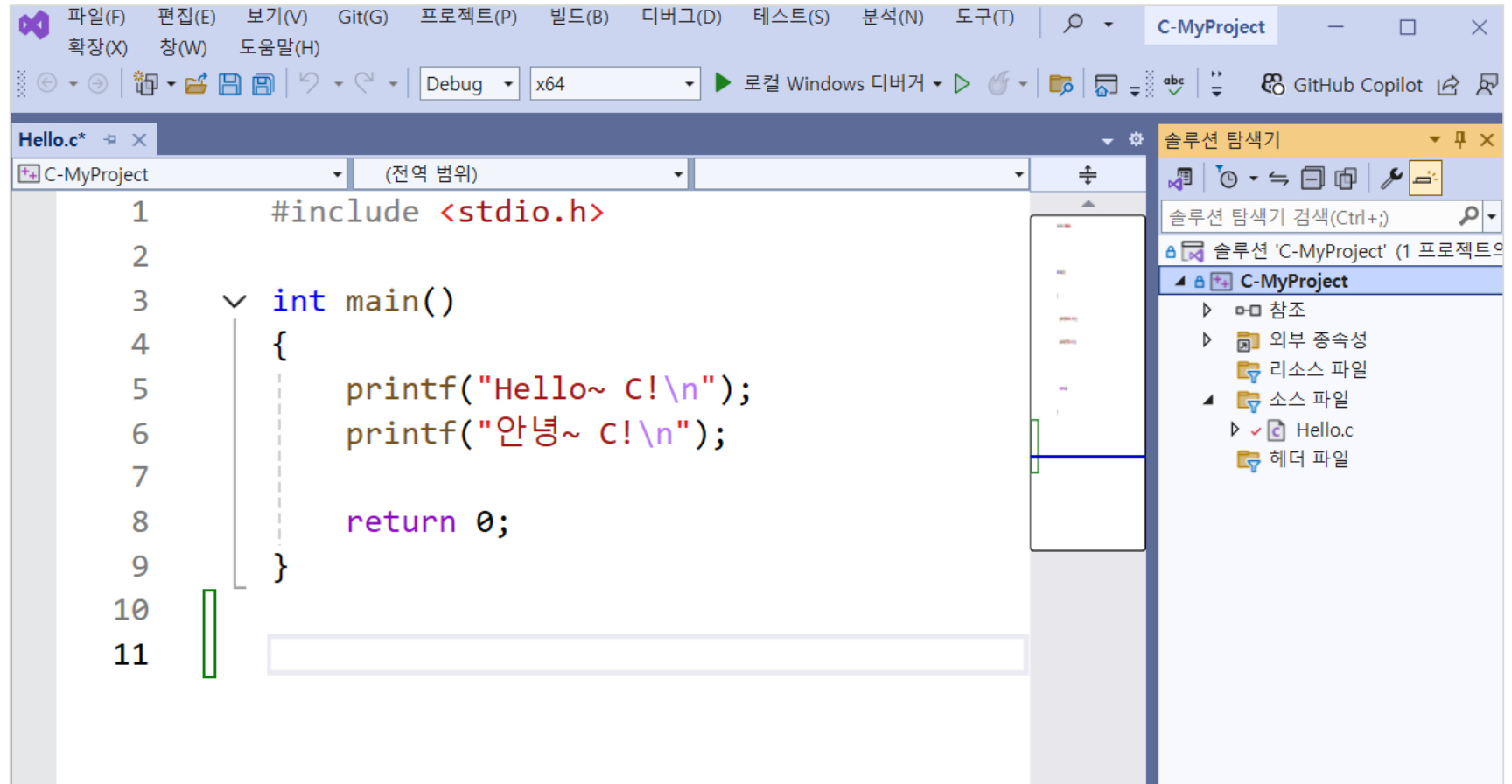
◆ 파일(.c) 만들기

소스파일 우클릭 -> 추가 -> 새항목 -> hello.c



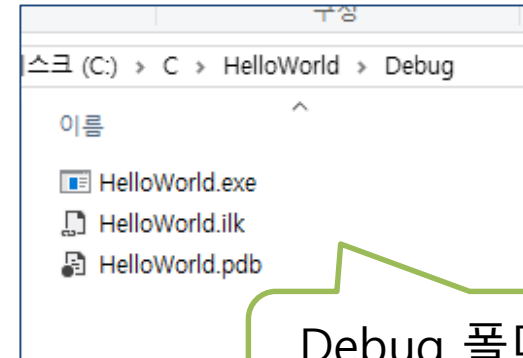
통합개발환경(IDE) – 비주얼 스튜디오

◆ 코드 작성 및 콘솔 출력

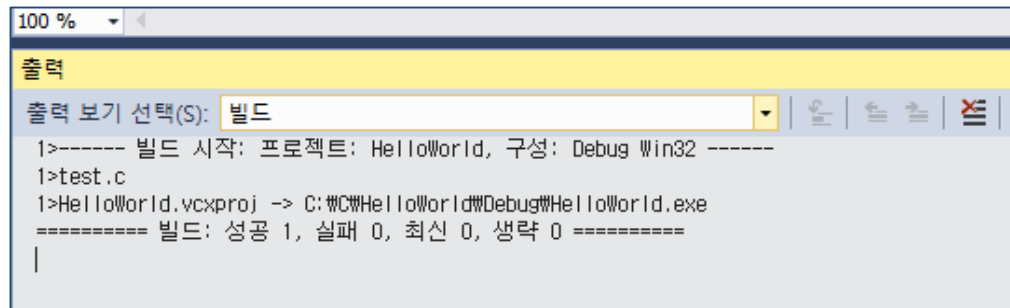


통합개발환경(IDE) – 비주얼 스튜디오

◆ 솔루션 빌드하기



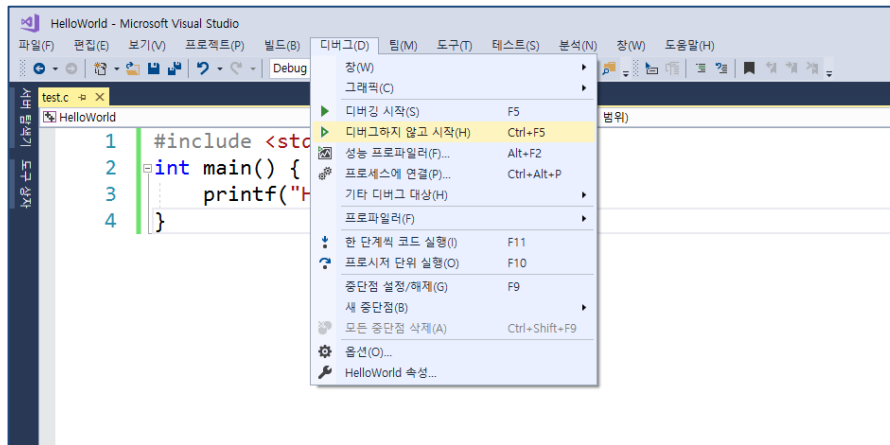
Debug 폴더가 생성,
실행 파일(.exe) 생성



통합개발환경(IDE) – 비주얼 스튜디오

◆ 파일 실행

디버그하지 않고 시작하기(단축키: Ctrl+F5)



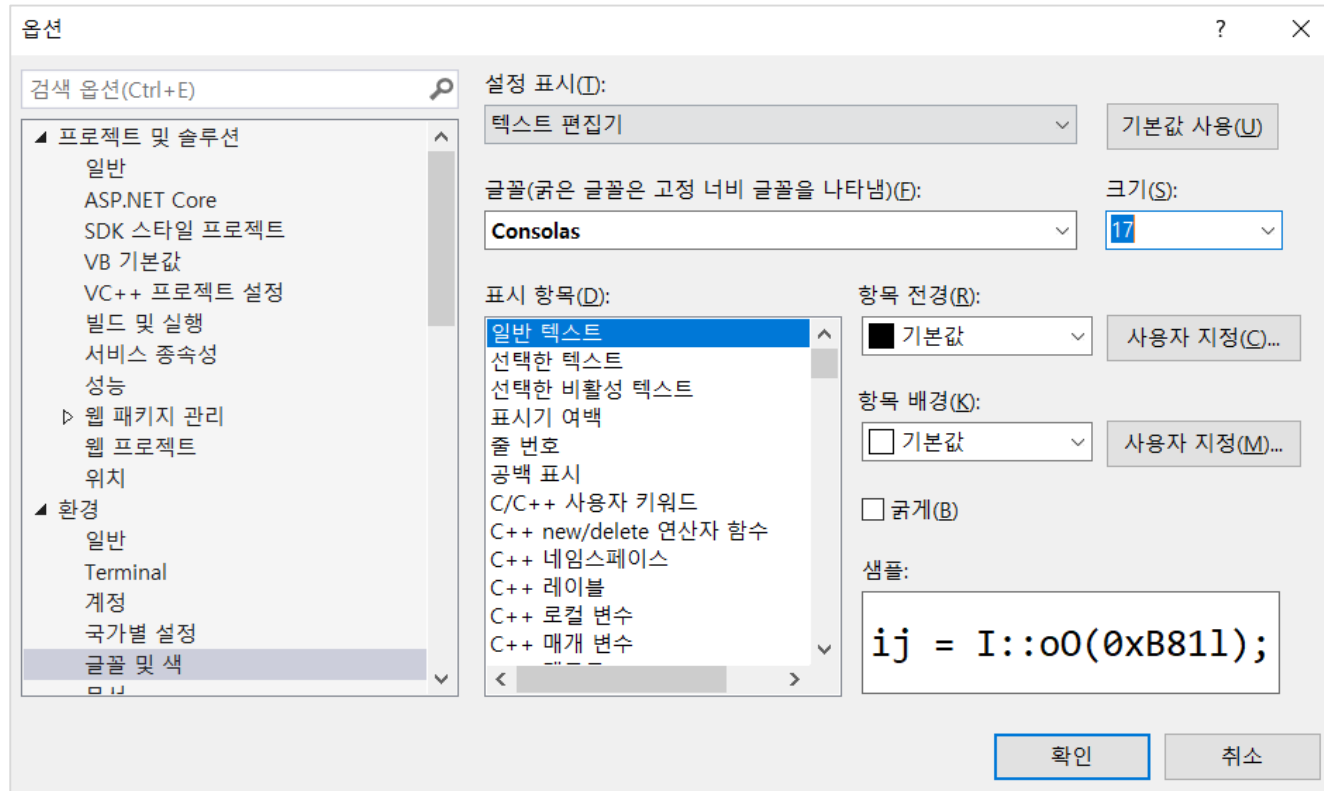
```
Hello~ C!  
안녕~ C!
```

```
D:\korea_IT\c2025\C-MyProject\x64\Debug\C-  
이 창을 닫으려면 아무 키나 누르세요 ...|
```



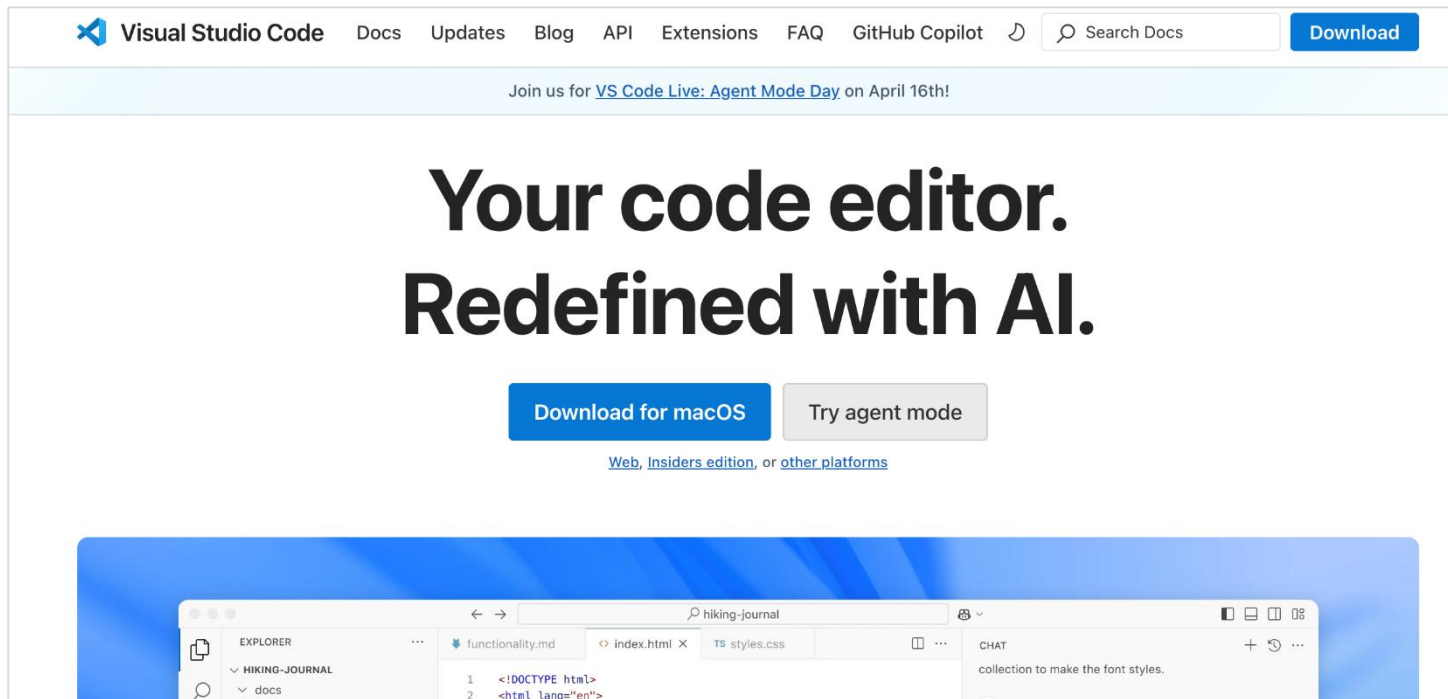
통합개발환경(IDE) – 비주얼 스튜디오

◆ 글꼴 설정 – 도구 > 옵션 > 환경 > 글꼴 및 색



Mac OS – 비주얼 스튜디오 코드

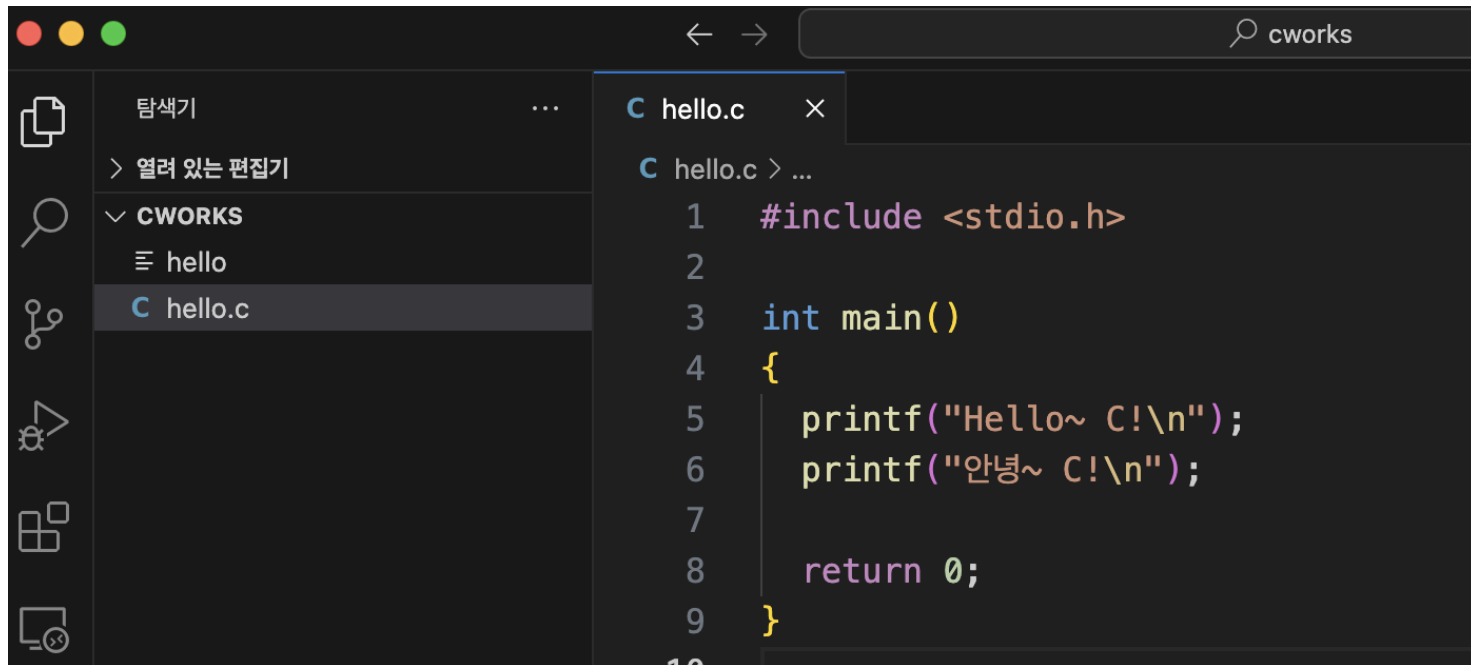
- ◆ 맥 OS에서 C, C++ 개발하기
 - macOS 용 다운로드



Mac OS – 비주얼 스튜디오 코드

◆ 맥 OS에서 C, C++ 개발하기

- 작업 폴더: cworks > hello.c 파일 > 코드 입력



The screenshot shows the Visual Studio Code interface on a Mac. The left sidebar displays the Explorer view with a folder named 'CWORKS' containing a subfolder 'hello' and a file 'hello.c'. The main editor area shows the code for 'hello.c'.

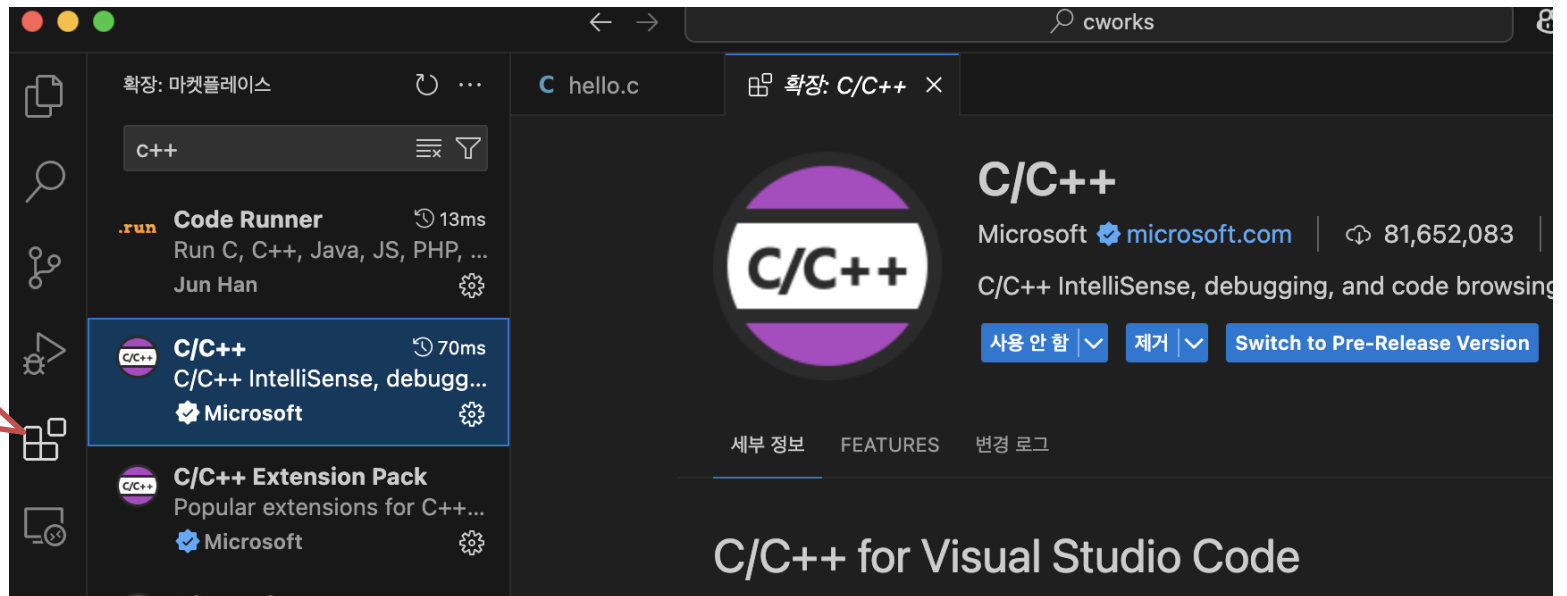
```
C hello.c
C hello.c > ...
1  #include <stdio.h>
2
3  int main()
4  {
5      printf("Hello~ C!\n");
6      printf("안녕~ C!\n");
7
8      return 0;
9  }
```



Mac OS – 비주얼 스튜디오 코드

◆ 맥 OS에서 C, C++ 개발하기

- ① 확장메뉴 – c,c++ 설치
- ② 확장메뉴 – Code Runner 설치



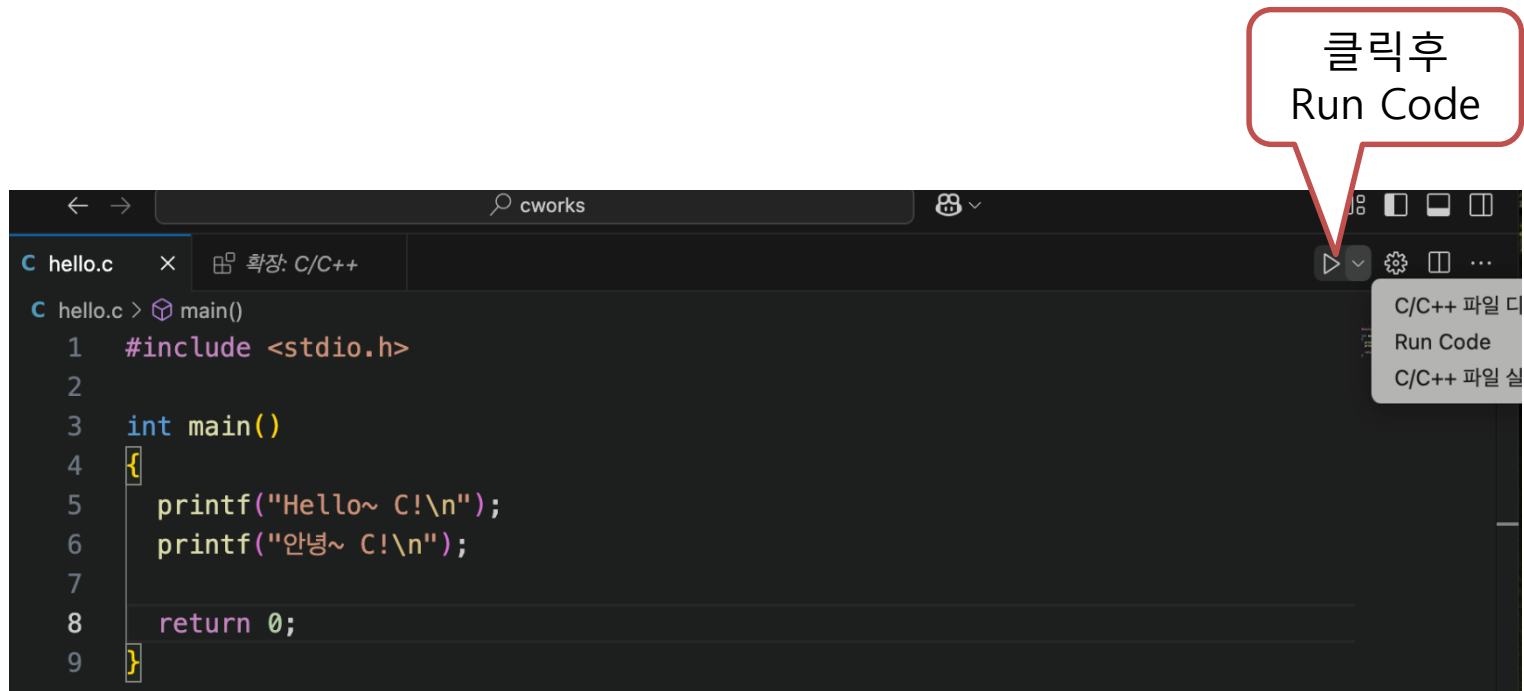
확장
메뉴



Mac OS – 비주얼 스튜디오 코드

◆ 맥 OS에서 C, C++ 개발하기

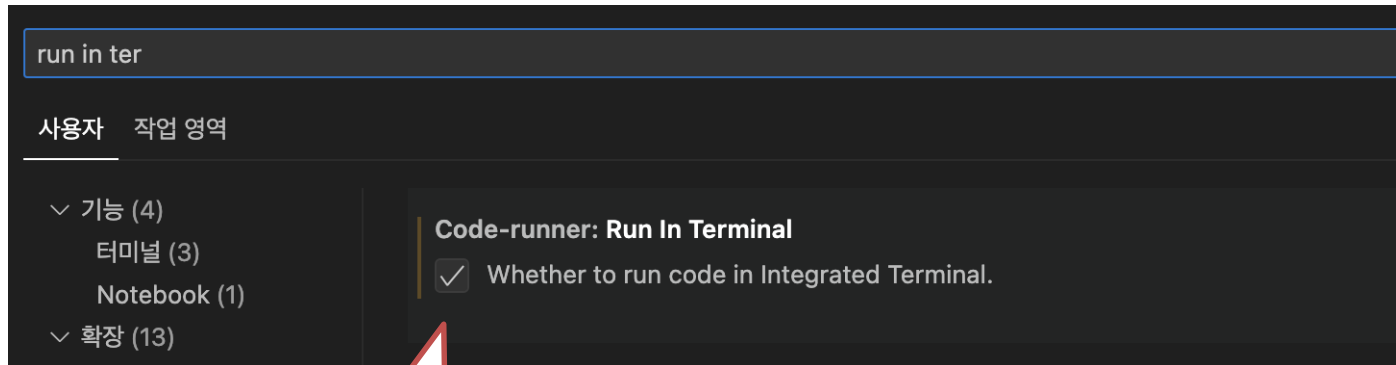
- 파일 실행(Run Code)



Mac OS – 비주얼 스튜디오 코드

◆ 맥 OS에서 C, C++ 개발하기

- Code > 기본설정 > 설정 > run in terminal 검색 > 체크



체크



C 프로그램의 구성 요소

◆ C 프로그램 소스 코드

- C 프로그램은 소스 코드로 이루어진 텍스트 파일이다. 확장자(*.c)
- **main()** 함수와 다양한 함수들로 구성되어 있다.

```
#include <stdio.h>

int main() {
    //문자
    printf("Hello~ C!\n");
    printf("안녕~ C!\n");

    //숫자
    printf("%d\n", 4);
    printf("%f\n", 2.54);

    return 0;
}
```

헤더파일 포함

세미콜론

주석

return 0은 운영체제로 값을 반환하여 프로그램을 종료한다는 의미



C언어의 기본 구조

◆ 전처리기와 헤더파일

① ② ③
`# include <stdio.h>`

① 전처리기(preprocessor)- # : 컴파일을 수행하기 전에 먼저 처리

② `include` : '포함하다' 라는 뜻을 가지며, 전처리를 지시

③ 헤더파일(Header File) : 확장자 `.h`를 가지는 파일

- `stdio` 의미 : **S**tandard **I**nput **O**utput (표준 입력 출력)

- `stdio.h` 의미 : **표준 입력 출력** 함수들을 가지고 있는 헤더 파일

- 설치된 경로

C:\Program Files\Microsoft VisualStudio\2022\Community\VC
\Tools\MSVC\14.42.34433\include



C 프로그램의 구성 요소

◆ { } 과 세미콜론

- 함수와 제어문은 { } 사이에 내용을 정의한다.
- 문자의 끝은 항상 세미콜론(;)으로 끝난다

◆ 주석문

- ① 주석(Comment) : 소스코드에 대한 설명
- ② 컴파일 되지 않는다.
- ③ 주석 처리 방법

```
/*  
파일명: Hello.c  
만든이: 홍길동  
프로그램 내용: Hello C world 테스트  
*/
```

↑
여러 줄 주석 처리

```
// 파일명: Hello.c  
// 만든이: 홍길동  
// 프로그램 내용: Hello C world 테스트
```

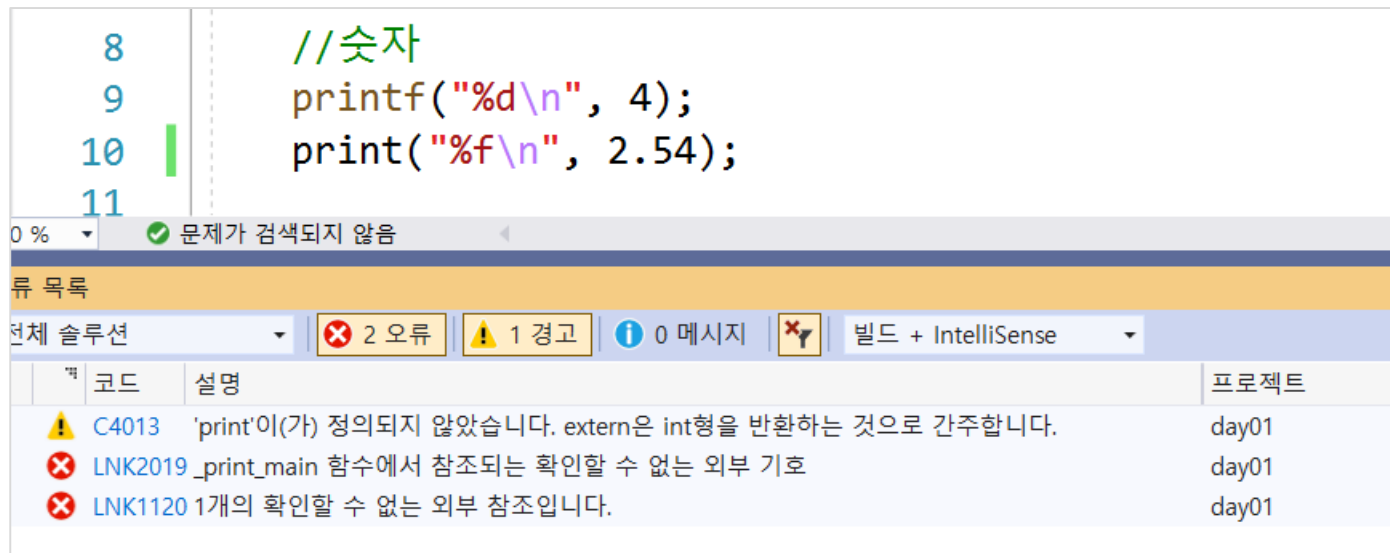
↑
한 줄 주석 처리



모니터에 데이터 출력 – printf() 함수

◆ 디버깅

컴파일 과정에서 오류가 발생한 것을 버그(bug)라 하고, 오류를 수정하는 작업을 디버그(debug) 또는 디버깅(debuging) 과정이라고 한다.



The screenshot shows a C++ IDE with the following code:

```
8 //숫자
9 printf("%d\n", 4);
10 print("%f\n", 2.54);
11
```

Below the code, the status bar indicates "0 %", "문제가 검색되지 않음", and "빌드 + IntelliSense".

The error list shows the following messages:

코드	설명	프로젝트
C4013	'print'이(가) 정의되지 않았습니다. extern은 int형을 반환하는 것으로 간주합니다.	day01
LNK2019	_print_main 함수에서 참조되는 확인할 수 없는 외부 기호	day01
LNK1120	1개의 확인할 수 없는 외부 참조입니다.	day01



모니터에 데이터 출력 – printf() 함수

- 화면 출력 함수 printf()

1. **printf**(출력 데이터);

큰 따옴표("")로 묶여있는 문자열만 출력

ex) printf("안녕하세요");

2. **printf**("출력 형식", 출력 데이터);

원하는 데이터(정수, 문자)를 출력

```
#include <stdio.h>

int main() {
    //문자
    printf("Hello~ C!\n");
    printf("안녕~ C!\n");

    //숫자
    printf("%d\n", 4);
    printf("%f\n", 2.54);

    return 0;
}
```



입.출력 서식 문자

■ 출력 및 입력 서식 및 제어 문자

서식문자	형태
%d	10진수 정수
%x	16진수 정수
%f	10진수 실수
%lf	10진수 실수
%c	한 개의 문자
%s	문자열 출력

특수문자	설명
\a	'뵁' 하는 경고음 소리 발생
\n	줄바꿈(개행)
\r	동일한 줄의 첫번째 위치로 커서 이동
\t	탭



Git - 소스 코드 관리



GitHub



깃허브(Git Hurb)

■ 깃허브란?

분산 버전 관리 툴인 깃 저장소 호스팅을 지원하는 웹 서비스이다.

깃을 창시한 사람은 리눅스를 만든 리누즈 토발즈이고, 깃허브를 인수하여 운영하는 곳은 마이크로소프트(MS)사이다.

■ 깃허브 환경 구축

1. 깃 소프트웨어 설치(git-scm.com)
2. 깃허브 가입(github.com) 및 원격 저장소 생성
3. 명령 프롬프트 사용(CLI 프로그램)



깃 소프트웨어 설치

■ Git – 소프트웨어 설치

git-scm.com > 다운로드 후 설치 > 계속 next



Download for Windows

[Click here to download](#) the latest (2.34.1) 64-bit version of the recent maintained build. It was released about 1 month ago.

Other Git for Windows downloads

Standalone Installer

[32-bit Git for Windows Setup.](#)

[64-bit Git for Windows Setup.](#)



깃허브 원격 저장소 만들기

■ 깃허브 가입하기

Sign Up > 메일로 코드 확인

Welcome to GitHub!
Let's begin the adventure

Enter your email
✓ sugu2000kr@naver.com


Create a password
✓

Enter a username
✓ sugu2000kr

Would you like to receive product updates and announcements via email?
Type "y" for yes or "n" for no
→ y

Continue

Here's your GitHub launch code, @sugu2000kr!



Continue signing up for GitHub by entering the code below:

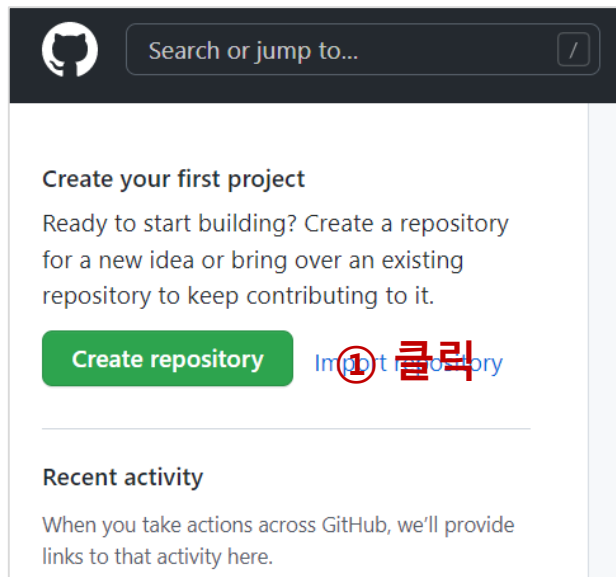
93221781

Open GitHub



깃허브 원격 저장소 만들기

Repository(저장소) 만들기



Search or jump to...

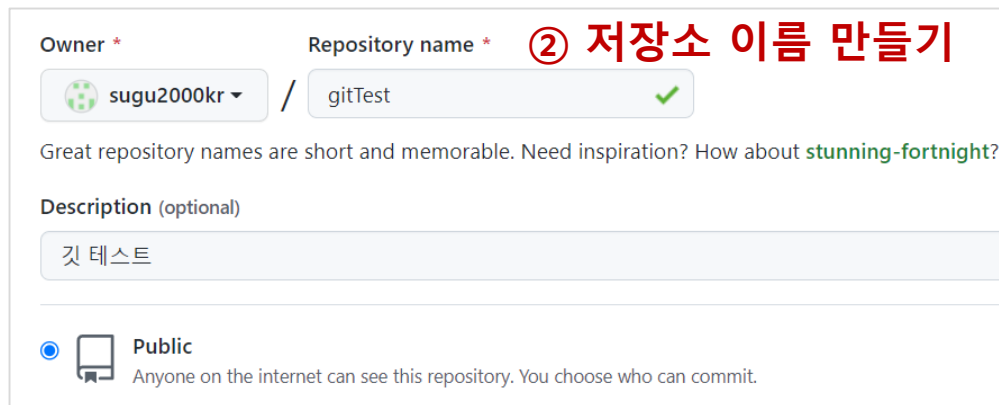
Create your first project

Ready to start building? Create a repository for a new idea or bring over an existing repository to keep contributing to it.

Create repository Import repository

Recent activity

When you take actions across GitHub, we'll provide links to that activity here.



Owner * Repository name * ② 저장소 이름 만들기

sugu2000kr / gitTest

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

Description (optional)

깃 테스트

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

③ 깃 명령어

Quick setup — if you've done this kind of thing before

 Set up in Desktop or ☐ HTTPS ☐ SSH <https://github.com/sugu2000kr/gitTest.git>

Get started by [creating a new file](#) or [uploading an existing file](#). We recommend every repository in

...or create a new repository on the command line

```
echo "# gitTest" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/sugu2000kr/gitTest.git
git push -u origin main
```



명령 프롬프트 사용

■ 깃허브 사용 툴 - 명령 프롬프트

* 윈도우 - 검색 - cmd - 명령 프롬프트

C:W>git

C:W>git -version

* 사용자 확인

C:W>git config user.name

```
C:\Users\김기용>git
usage: git [--version] [--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>]
        <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
    sparse-checkout    Initialize and modify the sparse-checkout


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
```



깃 환경 설정

■ Git 초기 환경 설정

git config 명령은 컴퓨터 1대에서 처음 한번만 실행함

C:\WgitTest> git config --user.name //git 계정확인

C:\W gitTest > git config --global user.name "kiyongee2"(본인 ID)

C:\W gitTest > git config --global user.email "kiyongee2@gmail.com"

C:\W gitTest> git init #git 초기화하기



깃에 파일 업로드하기

■ 깃에 파일 업로드하기 – 처음 업로드시

> git **status** (상태 확인)

➤ git add hello.txt (파일 1개업로드시)
git add . (모든 파일 add * 도 가능) //git 추가하기

> git **commit -m** "Add hello.txt" //커밋

> git **remote add origin** http://github.com/kiyongee2/gitTest.git

> git **push -u** origin master



깃에 파일 업로드하기

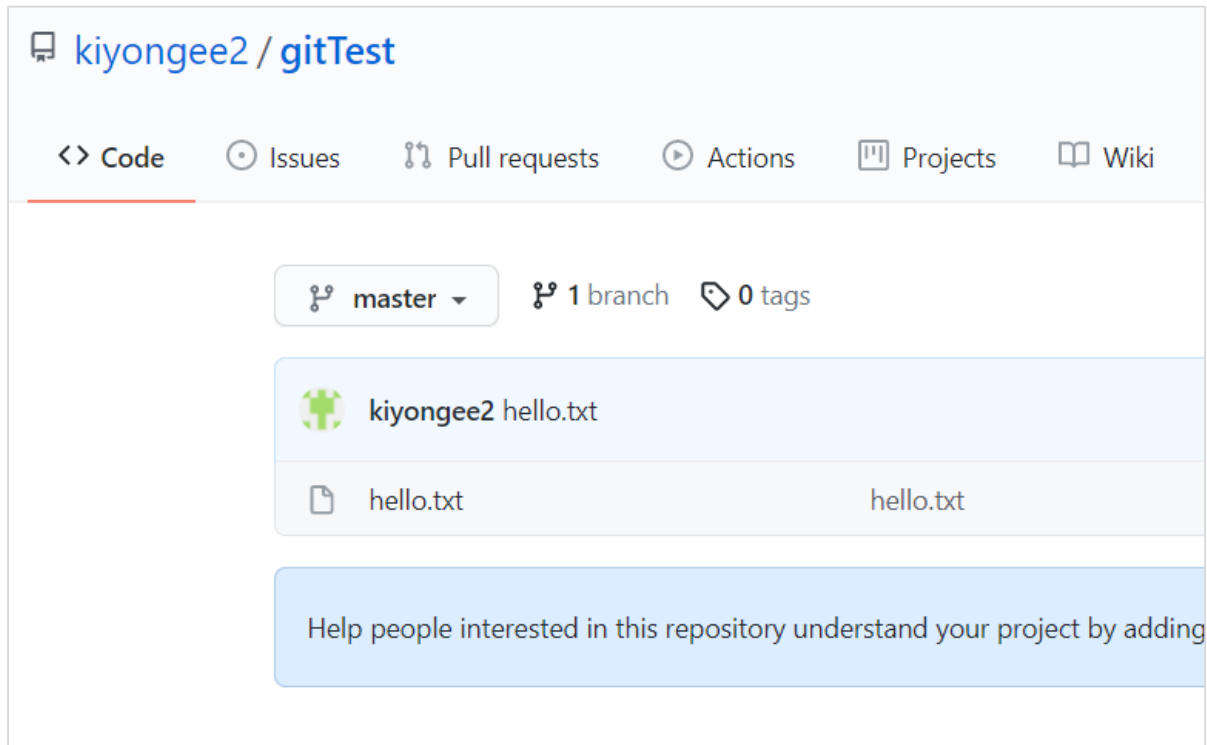
■ 깃에 파일 업로드하기 – 두번째 이후

- > git **status** 상태 확인
- > git **add** *
- > git **commit** -m "Add 추가 파일"
- > git **push**



깃허브 레포지터리 보기

■ 업로드된 파일 확인하기



깃 파일 삭제

■ 파일 삭제하기

>git **rm** 파일이름

>git **commit -m** "Delete 파일이름"

>git **push**

■ 디렉터리 삭제하기

>git **rm -rf** 디렉터리 이름

>git **commit -m** "Delete 디렉터리 이름"

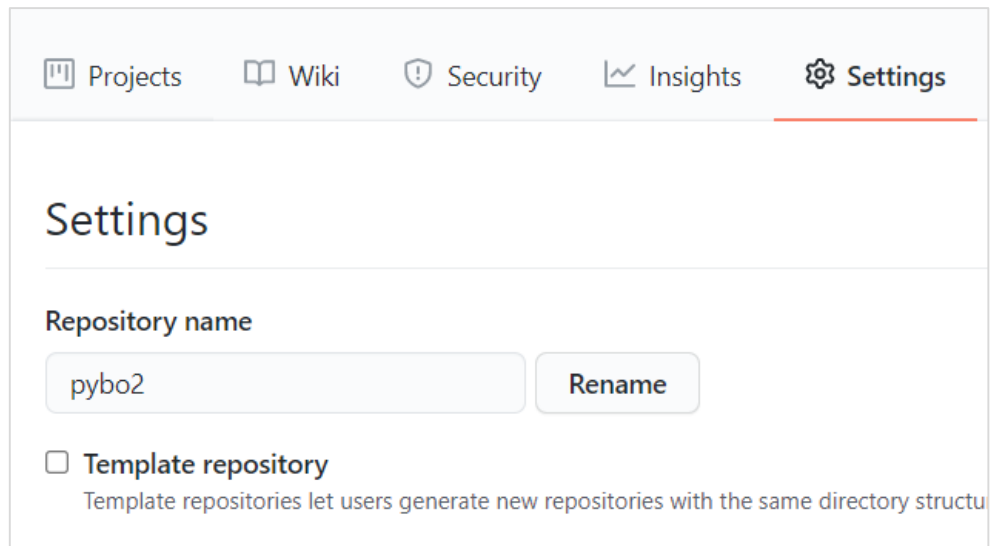
>git **push**



깃 계정 이름 변경

■ 계정 이름 변경하기

Settings > 변경할 이름 > Rename



The screenshot shows the GitHub 'Settings' page for a repository. The navigation bar at the top includes 'Projects', 'Wiki', 'Security', 'Insights', and 'Settings' (which is highlighted with a red underline). Below the navigation bar, the 'Settings' title is displayed. Under the 'Repository name' section, there is a text input field containing 'pybo2' and a 'Rename' button. Below this, there is a checkbox labeled 'Template repository' which is currently unchecked. A descriptive text below the checkbox states: 'Template repositories let users generate new repositories with the same directory structure'.



깃 계정 삭제

■ 계정 삭제하기

Settings > Danger Zone

Are you absolutely sure?

×

Unexpected bad things will happen if you don't read this!

This action **cannot** be undone. This will permanently delete the **kiyongee2/gitTest** repository, wiki, issues, comments, packages, secrets, workflow runs, and remove all collaborator associations.

Please type **kiyongee2/gitTest** to confirm.


kiyongee2/gitTest

I understand the consequences, delete this repository




내 컴퓨터의 다른 사용자 계정 삭제하기


❖ 이미 사용중인 다른 사용자 계정 삭제하기

 > 제어판 > 사용자 계정 > 자격 증명 관리자


자격 증명 관리

웹 사이트, 연결된 응용 프로그램 및 네트워크에 대해 저장된 로그인 정보를 보고 삭제합니다.

 웹 자격 증명

 Windows 자격 증명

git:https://github.com

수정한 날짜: 오늘 

인터넷 또는 네트워크 주소: git:https://github.com

사용자 이름: sugu2100

암호:

지속성: 로컬 컴퓨터

[편집](#) [제거](#)

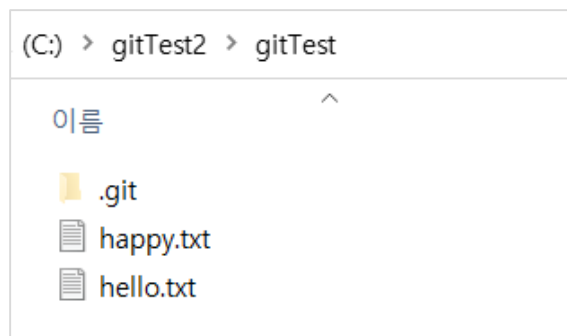


깃 클론(git clone)

- 원격저장소에서 자료 가져오기

처음엔 `git clone` > 2번째 부터 `git pull` 사용

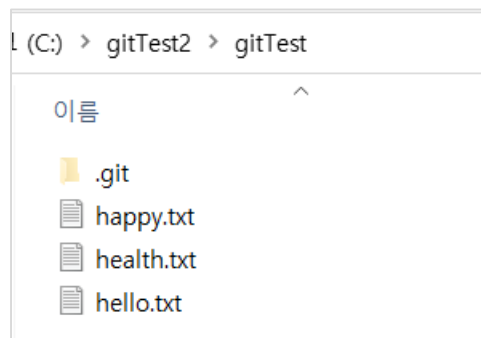
`c:\WgitTest2>git clone https://github.com/kiyongee2/gitTest`



gitTest에서
health.txt - 업로드

2번째 부터 추가 파일이 있는 경우

`c:\WgitTest2>git pull`



브랜치 이름 변경하기

■ 브랜치 master -> main으로 변경

개발을 하다 보면 코드를 여러 개로 복사해야 하는 일이 자주 생긴다.

코드를 통째로 복사하고 나서 원래 코드와는 상관없이 독립적으로 개발을 진행할 수 있는데, 이렇게 독립적으로 개발하는 것이 브랜치다.

```
c:\WgitTest>git branch
```

```
*master
```

```
c:\WgitTest>git branch -M main
```

```
*main
```

```
c:\WgitTest>git push -u origin main
```

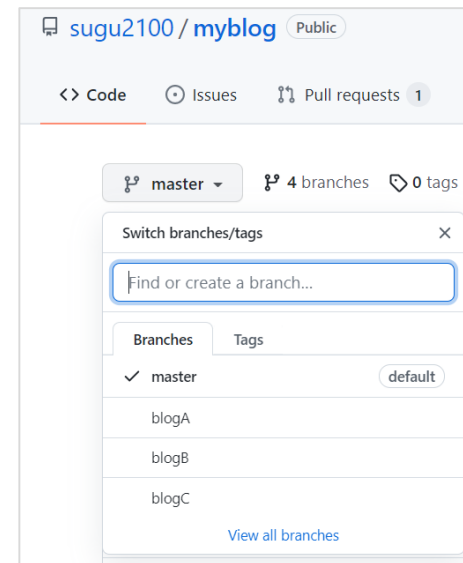


새 브랜치 만들기

■ 새 브랜치 만들기

1. 새 브랜치 만들기 - **git branch** 브랜치 이름

```
c:\WgitTest>git branch blogA  
c:\WgitTest>git branch  
  
*master  
  
blogA
```



2. blogA 원격 계정에 추가하기

```
c:\WgitTest>git remote add blogA https://github.com/sugu2100/myblog  
c:\WgitTest>git push blogA
```



브랜치 이동하기

- 브랜치 이동하기

blogA로 브랜치 이동 – git checkout 브랜치 이름

```
c:\WgitTest>git checkout blogA
c:\WgitTest>git branch
  master
* blogA
```

- 자료 수정후 깃에 업로드하기

```
c:\WgitTest>git add *
C:\WgitTest>git commit -m "추가"
C:\WgitTest>git push blogA
```

