

C++_개발 환경 구축

Visual Studio 2022

C언어와 C++ 언어 비교

● C 언어

- **Ken Thompson, Dennis Ritchie**
- AT & T 벨연구소, 1970년
- 유닉스(UNIX) 운영체제에서 사용하기 위해 만든 언어
- 고급언어이지만 메모리를 직접 접근하는 저급언어 특징, 운영체제나 시스템프로그래밍에 적합하고, 하드웨어나 임베디드 시스템에서 많이 사용됨
- **절차적 프로그래밍** 기법
- 단점 : 다른 고급언어에 비해 이해가 쉽지 않고, 개발 기간이 오래 걸리며, 대형 프로젝트에 적합하지 않다.

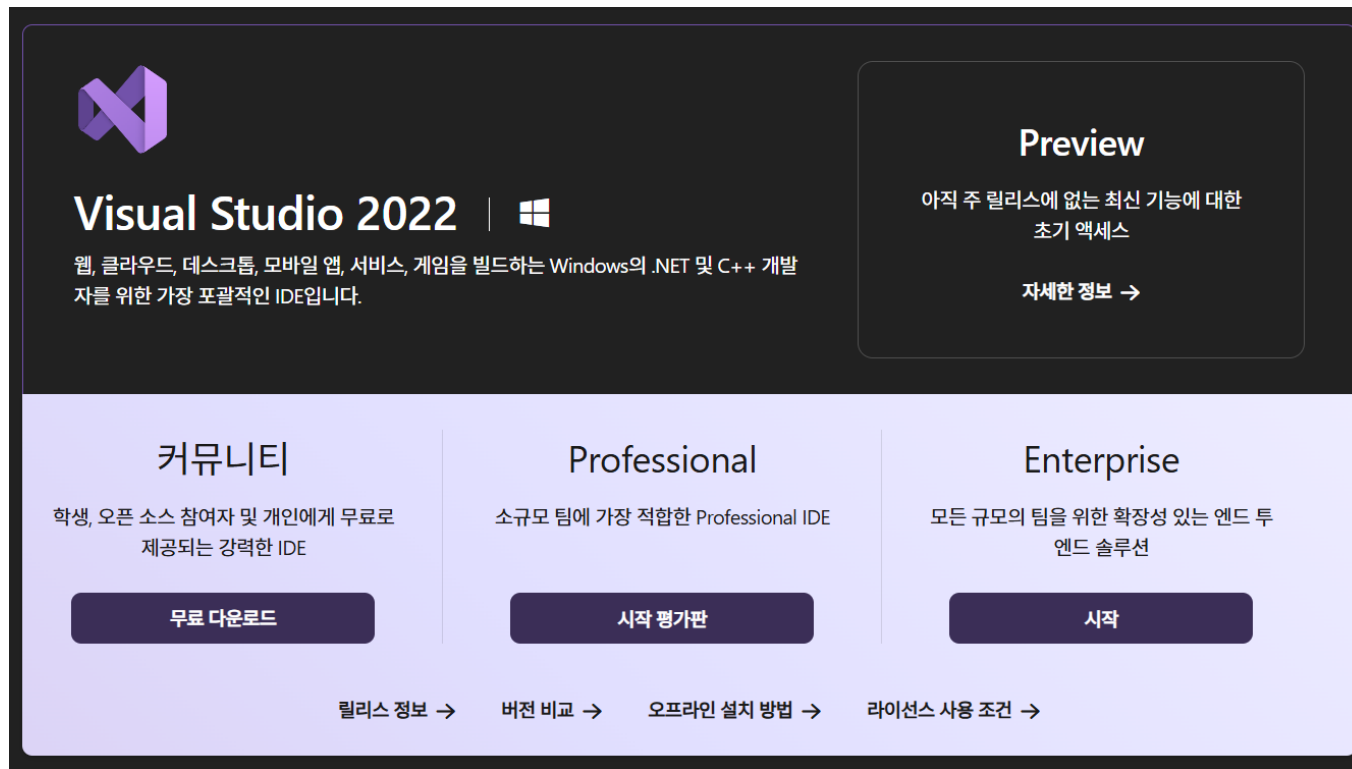
C언어와 C++ 언어 비교

● C++ 언어


- **Bjarne Stroustrup(비야네 스트루스트룹)**
- AT & T 벨연구소, 1983년
- C언어를 확장한 프로그래밍 언어
- 절차적프로그래밍 기법을 지원하면서 추가적으로 **객체지향프로그래밍 기법**
- 캡슐화, 정보은닉, 상속과 다형성
- 템플릿(template)을 통해 일반화프로그래밍 기법 제공
- 대규모 소프트웨어 개발에 적합하다.(은행 업무, 게임, 임베디드 프로젝트 등)


통합개발환경(IDE) – 비주얼 스튜디오

- Visual Studio 다운로드
 - Community(커뮤니티) 2022 다운로드



The image shows the Visual Studio 2022 download page. At the top left is the Visual Studio logo. Next to it is the text 'Visual Studio 2022' followed by a Windows logo. Below this is a description: '웹, 클라우드, 데스크톱, 모바일 앱, 서비스, 게임을 빌드하는 Windows의 .NET 및 C++ 개발자를 위한 가장 포괄적인 IDE입니다.' To the right of this is a 'Preview' section with the text '아직 주 릴리스에 없는 최신 기능에 대한 초기 액세스' and a link '자세한 정보 →'. Below these are three columns for different editions: '커뮤니티' (Community), 'Professional', and 'Enterprise'. Each column has a description and a button. At the bottom, there are links for '릴리스 정보 →', '버전 비교 →', '오프라인 설치 방법 →', and '라이선스 사용 조건 →'.



Visual Studio 2022 | 

웹, 클라우드, 데스크톱, 모바일 앱, 서비스, 게임을 빌드하는 Windows의 .NET 및 C++ 개발자를 위한 가장 포괄적인 IDE입니다.

Preview

아직 주 릴리스에 없는 최신 기능에 대한 초기 액세스

[자세한 정보 →](#)

커뮤니티	Professional	Enterprise
학생, 오픈 소스 참여자 및 개인에게 무료로 제공되는 강력한 IDE	소규모 팀에 가장 적합한 Professional IDE	모든 규모의 팀을 위한 확장성 있는 엔드 투 엔드 솔루션
무료 다운로드	시작 평가판	시작

[릴리스 정보 →](#) [버전 비교 →](#) [오프라인 설치 방법 →](#) [라이선스 사용 조건 →](#)

통합개발환경(IDE) – 비주얼 스튜디오

● 솔루션 > 프로젝트 > 파일(.cpp)

- 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.

프로젝트 이름(I)

BasicC++

위치(L)

D:\korea_JTWc++\works\

솔루션(S)

새 솔루션 만들기

솔루션 이름(M) ⓘ

BasicC++

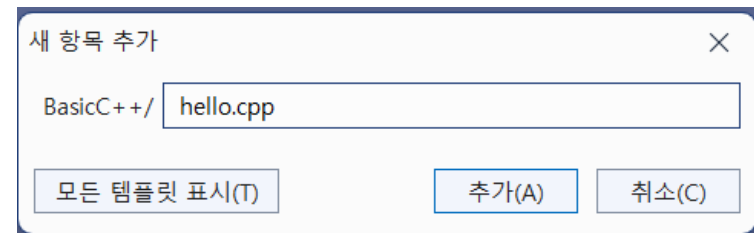
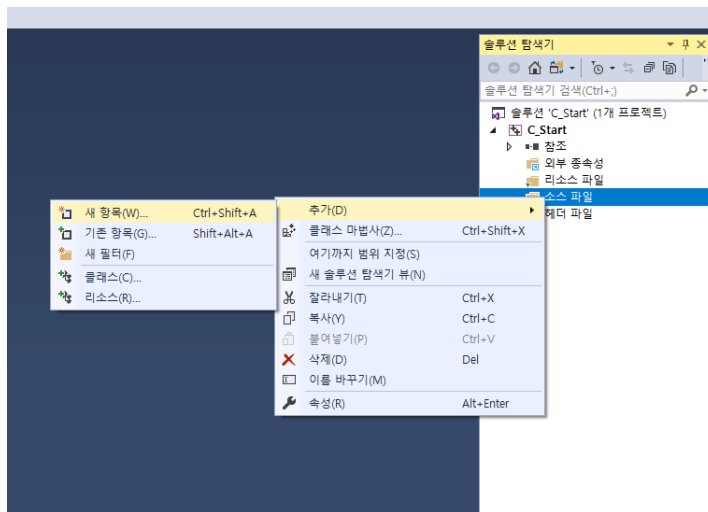
☐ 솔루션 및 프로젝트를 같은 디렉터리에 배치(D)

저장 경로 설정

통합개발환경(IDE) – 비주얼 스튜디오

● 파일(.cpp) 만들기

소스파일 우클릭 -> 추가 -> 새항목 -> hello.cpp



통합개발환경(IDE) – 비주얼 스튜디오

- 코드 작성 및 콘솔 출력

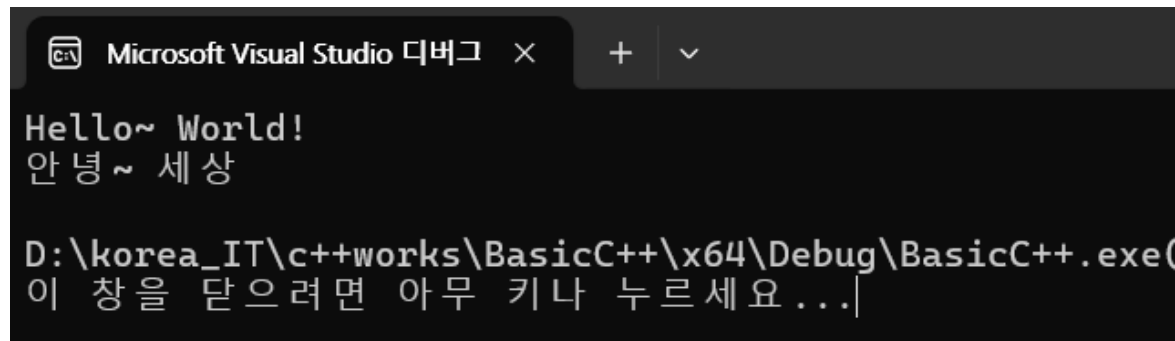
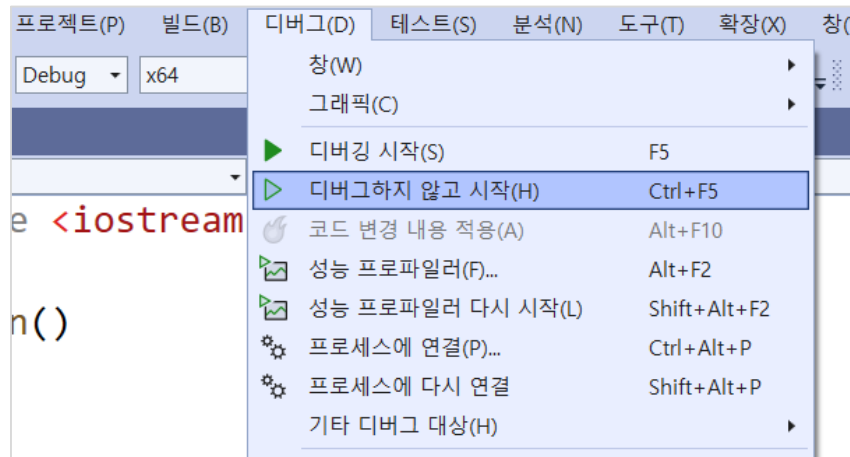
```
#include <iostream>

int main()
{
    std::cout << "Hello~ World!" << std::endl;
    std::cout << "안녕~ 세상" << std::endl;

    return 0;
}
```

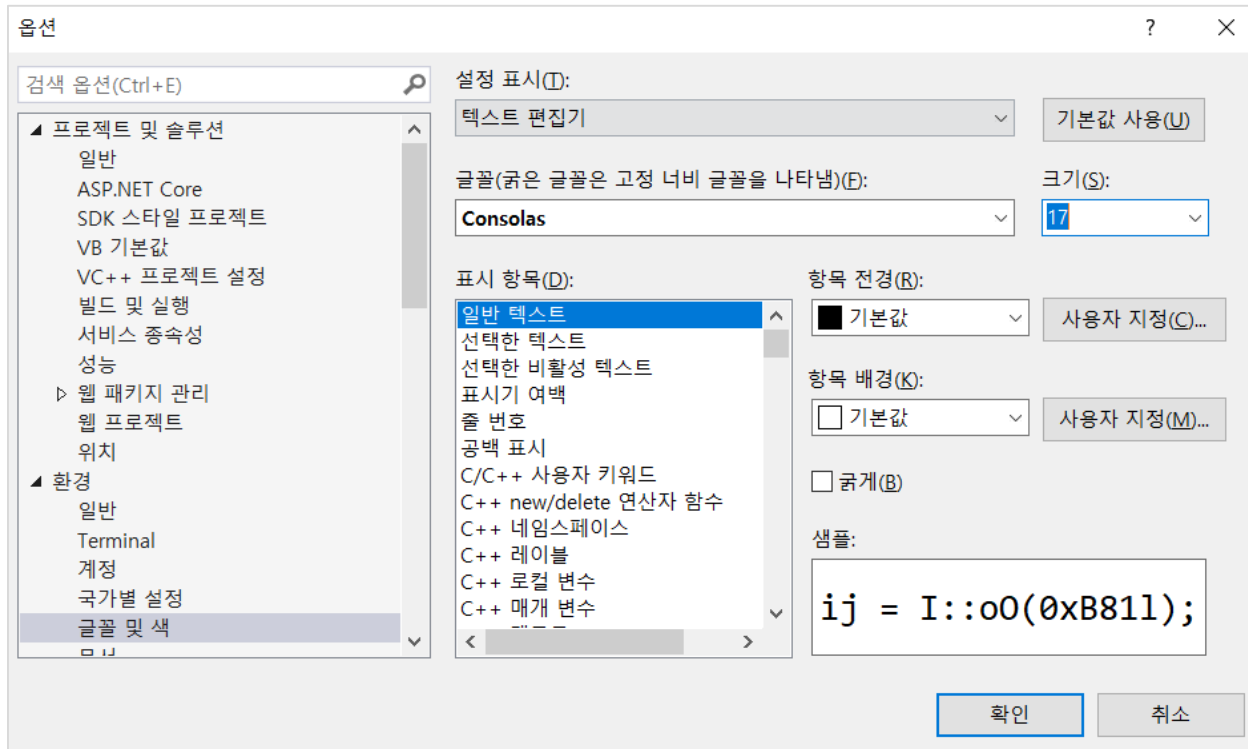
통합개발환경(IDE) – 비주얼 스튜디오

- 파일 실행 -> 디버그하지 않고 시작하기(Ctrl+F5)

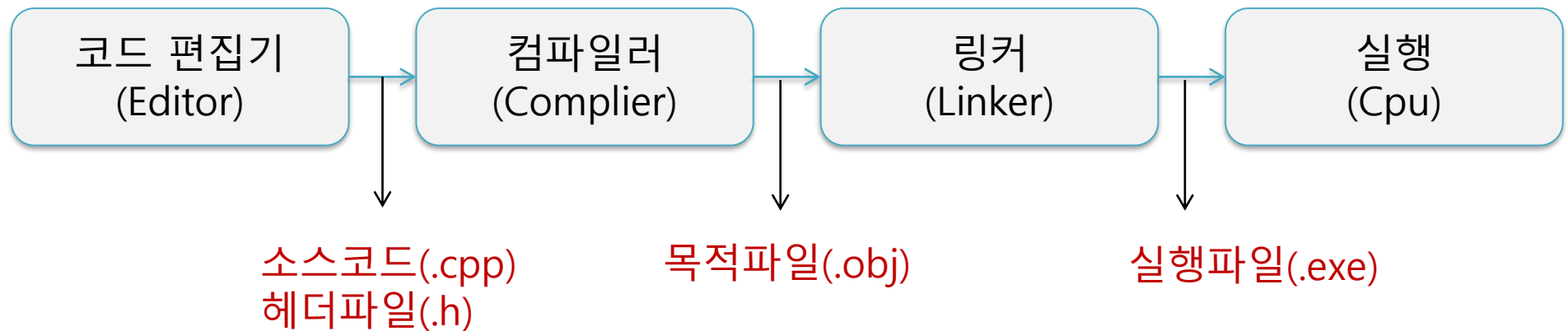


통합개발환경(IDE) – 비주얼 스튜디오

- 글꼴 설정 - 도구 > 옵션 > 환경 > 글꼴 및 색



C++ 프로그래밍 단계



➤ 빌드(build) 과정

- 전처리(preprocessing) : #include(지시자), #define 등 실제값으로 대체하여 컴파일을 위한 최종 소스파일을 만드는 과정
- 컴파일(compile) : 소스파일을 기계어로 변환하여 목적파일을 만든다.
- 링크(link) : 실제로 외부(다른파일, 라이브러리)에 존재하는 함수나 전역 변수들을 찾아서 연결하는 역할

C 프로그램의 구성 요소

● 세미콜론

- 문자의 끝은 항상 세미콜론(;)으로 끝난다

● 주석문

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

```
/*  
파일명: hello.cpp  
만든이: 김기용  
프로그램: Hello~ World 테스트  
*/
```

↑
여러 줄 주석 처리

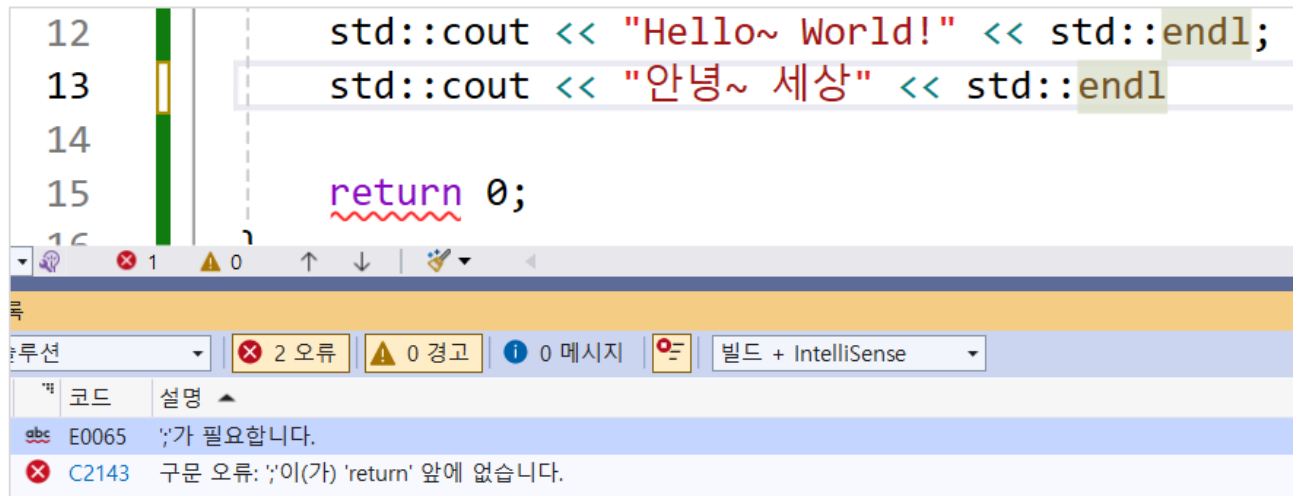
```
//std(클래스)는 namespace(이름 공간)  
//cout, endl은 함수  
std::cout << "Hello~ World!" << std::endl;  
std::cout << "안녕~ 세상" << std::endl;
```

↑
한 줄 주석 처리

디버깅(Debugging)

- 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++ code editor with the following code:

```
12 std::cout << "Hello~ World!" << std::endl;  
13 std::cout << "안녕~ 세상" << std::endl;  
14  
15 return 0;  
16
```

Below the code editor, the error list shows two errors:

- E0065 '가 필요합니다.
- C2143 구문 오류: '이(가) 'return' 앞에 없습니다.

〈iostream〉 헤더 파일

- C++ 표준 라이브러리 클래스

`cout`

표준 출력을 담당하는 ostream 클래스의 객체이다.

`cin`

표준 입력을 담당하는 istream 클래스의 객체이다.

`iostream`

입출력에 필요한 클래스와 전역 객체들을 정의한 헤더파일이다.

`<<` 연산자

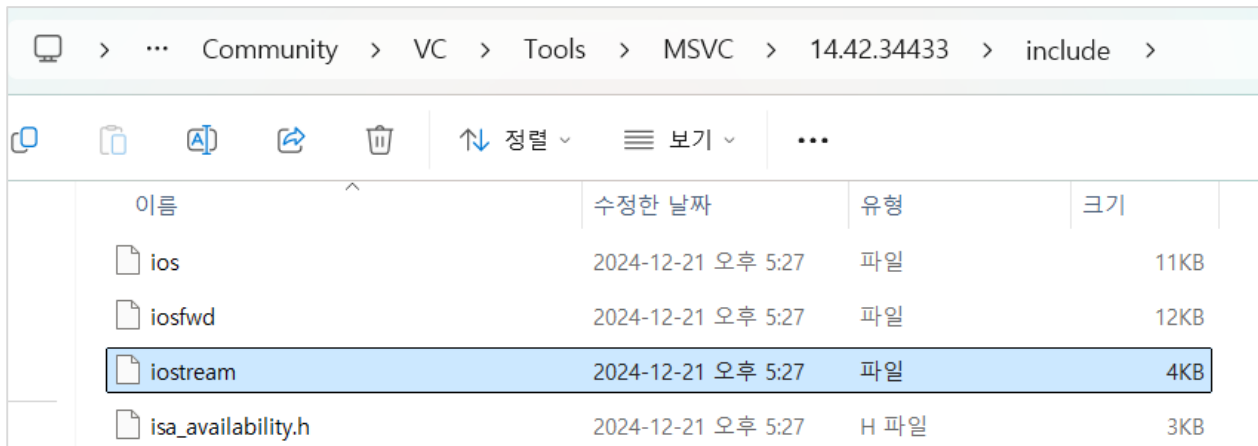
정수를 왼쪽으로 시프트(shift)하는 연산자.

오른쪽 피연산자 데이터를 왼쪽 스트림 객체에 삽입한다.

〈iostream〉 헤더 파일

- <iostream> 헤더 파일의 위치는?

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



이름	수정한 날짜	유형	크기
ios	2024-12-21 오후 5:27	파일	11KB
iosfwd	2024-12-21 오후 5:27	파일	12KB
iostream	2024-12-21 오후 5:27	파일	4KB
isa_availability.h	2024-12-21 오후 5:27	H 파일	3KB

실습 예제

● Hello.cpp

```
#include <iostream>
/*
파일명: hello.cpp
만든이: 김기용
프로그램: Hello~ World 테스트
*/

int main()
{
    //std(클래스)는 namespace(이름 공간)
    //cout, endl은 함수
    std::cout << "Hello~ World!" << std::endl;
    std::cout << "안녕~ 세상" << std::endl;

    //사칙 연산
    std::cout << 4 + 5 << std::endl;
    std::cout << 4 - 5 << std::endl;
    std::cout << 4 * 5 << std::endl;
    std::cout << 4.0 / 5.0 << std::endl;

    return 0;
}
```

C++_ Git(깃) 사용법

Visual Studio 2022

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!



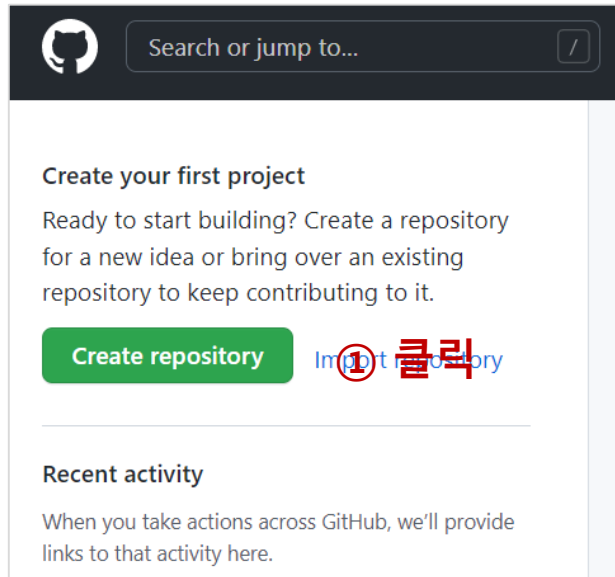
Continue signing up for GitHub by entering the code below:

93221781



Open GitHub

깃허브 원격 저장소 만들기

Repository(저장소) 만들기




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

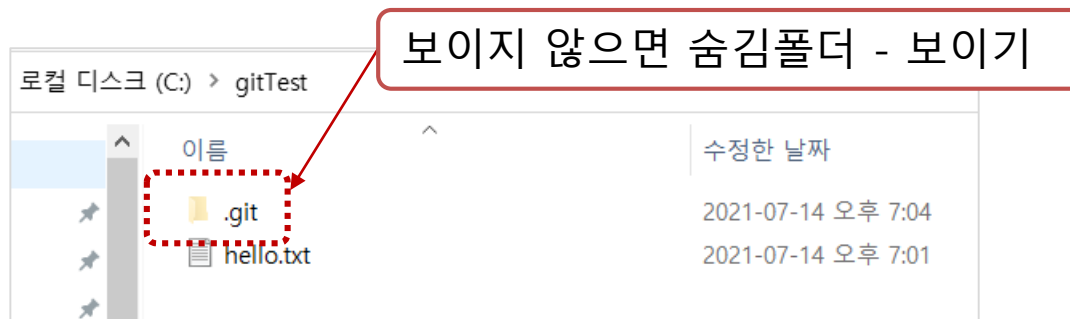
git config 명령은 컴퓨터 1대에서 처음 한번만 실행함

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

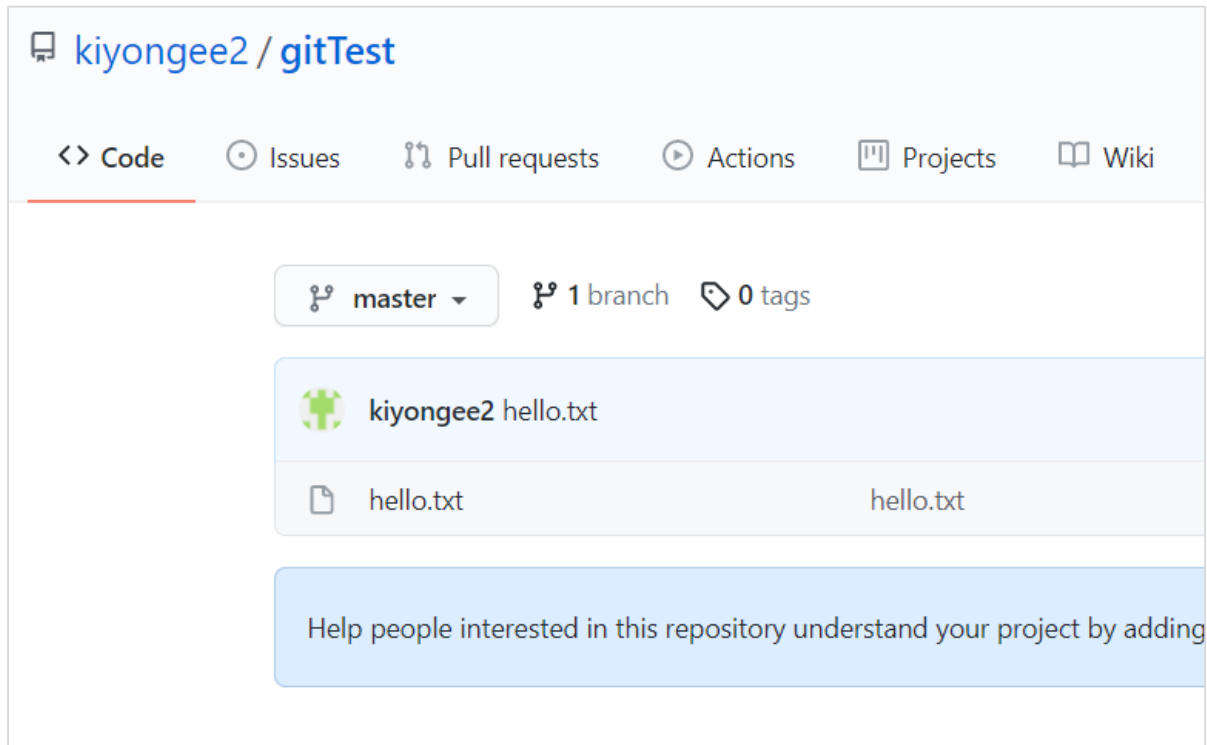
> git **add** *

> 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

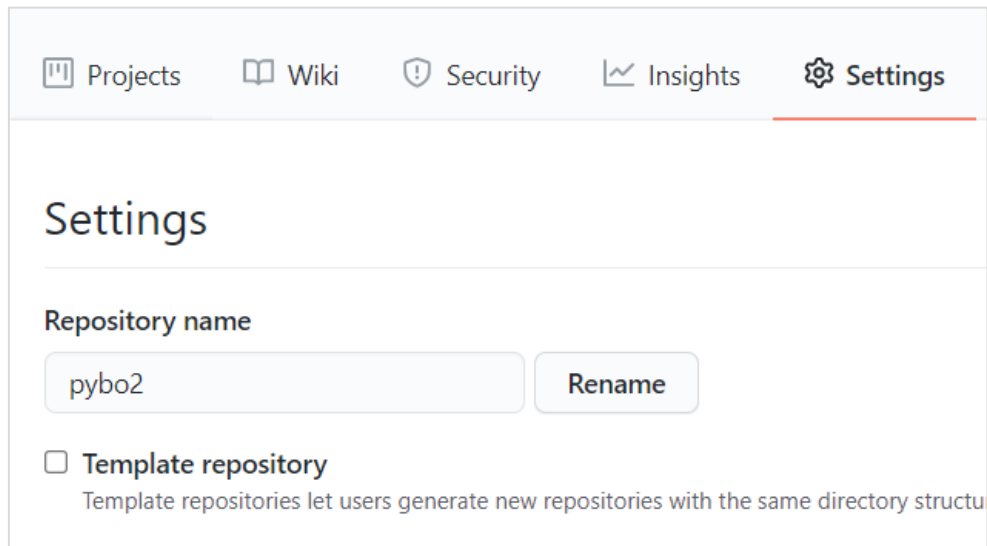
>git **commit -m** "Delete 디렉터리 이름"

>git **push**

깃 계정 이름 변경

■ 계정 이름 변경하기

Settings > 변경할 이름 > Rename



The screenshot shows the GitHub 'Settings' page. At the top, there is a navigation bar with icons and labels for 'Projects', 'Wiki', 'Security', 'Insights', and 'Settings'. The 'Settings' tab is selected and highlighted with a red underline. Below the navigation bar, the page title 'Settings' is displayed. Underneath, the 'Repository name' section is visible, featuring a text input field containing 'pybo2' and a 'Rename' button to its right. Below this, there is a checkbox labeled 'Template repository' which is currently unchecked. A descriptive text line follows: '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


깃 계정 삭제


■ 이미 사용중인 다른 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.com수정한 날짜: 오늘 

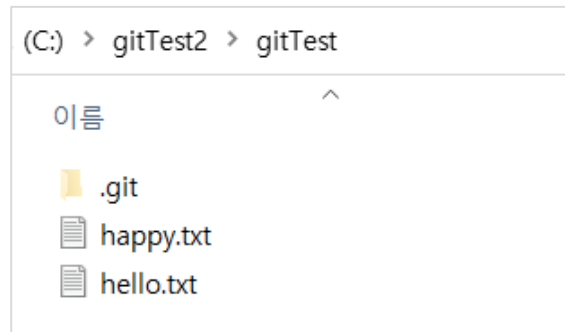
인터넷 또는 네트워크 주소: git:https://github.com
사용자 이름: sugu2100
암호:
지속성: 로컬 컴퓨터
[편집](#) [제거](#)

깃 클론(git clone)

- 원격저장소에서 자료 가져오기

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

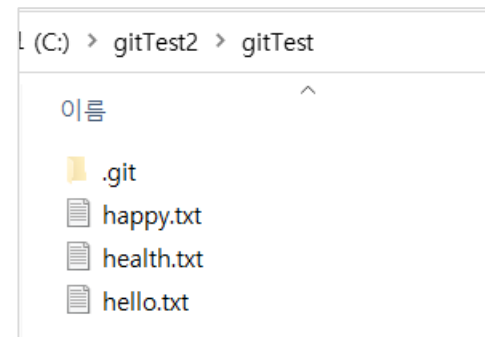
c:\WgitTest2> **git clone** https://github.com/kiyongee2/gitTest



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
```

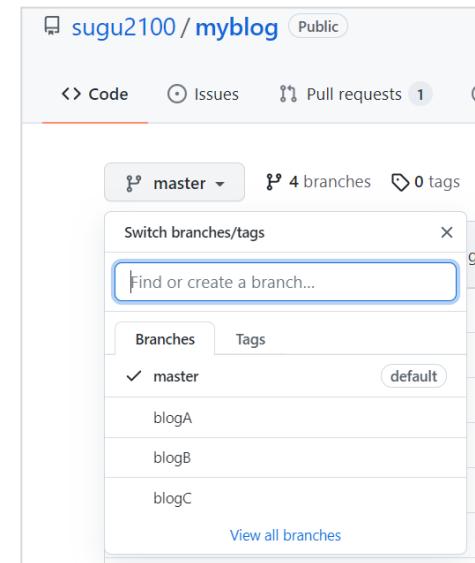
```
c:\WgitTest>git push -u origin main
```

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
```

실습문제 1 – 깃(git) 설치

- 실습 예제

깃(git) 소프트웨어 설치하고, 깃허브 리포지터리 계정에서 다운로드 받기

URL: https://github.com/kiyongee2/korea_it_java-.git
