



1-1e. (Mac용)프로그래밍 환경 구성

창의적소프트웨어프로그래밍
2022년도 여름학기
Racin
(NPC 무지성님이 만들어 주심)

이 슬라이드는

- 우리 수업 첫 날,
Windows 설치가 어려운 요즘 Macbook 쓰는 친구들을 위해 소개했던
Mac용 프로그래밍 환경 구성 방법을 요약해서 슬라이드로 만들었어요.
 - 제가 한 건 아니고 NPC 무지성 님이 만들어서 보내주심
- 일단 집에서 진행해 보고, 잘 안 되면 오픈톡방 등으로 물어봐 주세요.

순서

- Mac용 VS Code(Visual Studio Code) 다운로드 및 설치
- gcc 설치하기
- 작성한 .c / .cpp 파일로 프로그램 만드는 방법 소개
- 부록: VS Code 소개

VS Code 설치 (1/4)

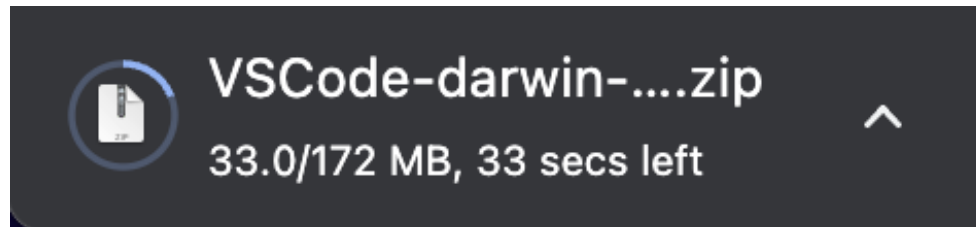
- 수업 페이지 자료실에서 아래와 같이 생긴 항목을 눌러 VSCode-darwin-universal.zip을 다운로드해 주세요:



(Mac용) VSCode-darwin-universal.zip

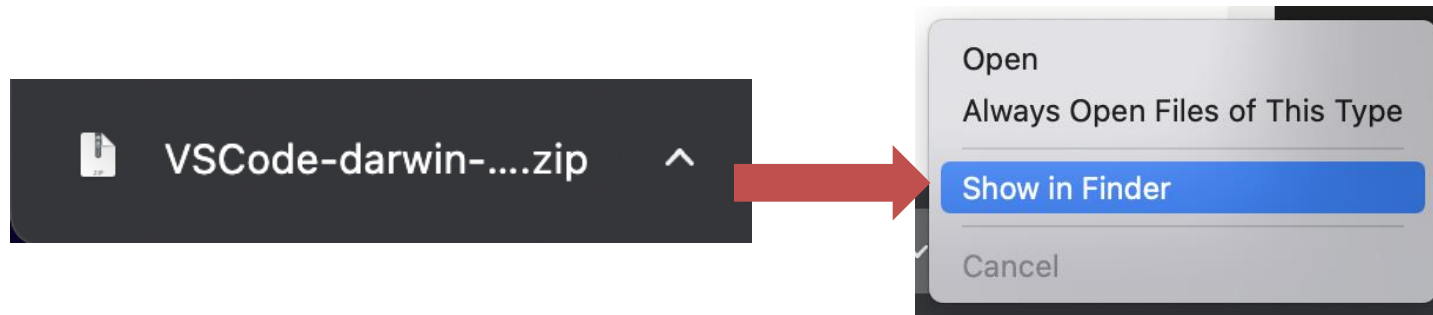
파일

- code.visualstudio.com 에서 직접 다운로드해도 되긴 해요
- 다운로드가 시작되면 아래와 같이 남은 시간을 알려줘요:



VS Code 설치 (2/4)

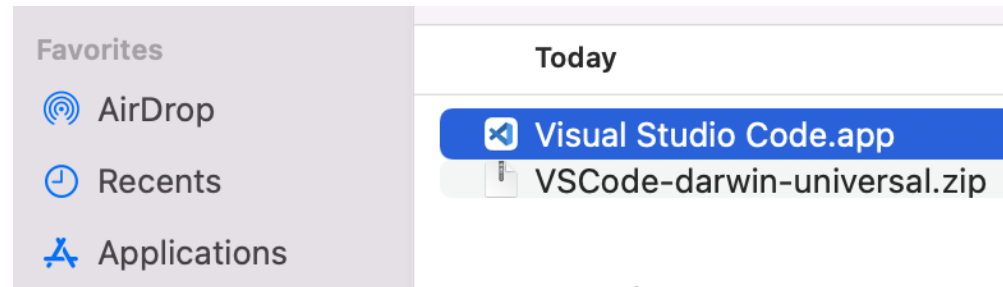
- 다운로드가 완료되었다면, 파인더에서 보기를 선택해 .zip 파일이 들어 있는 폴더를 열어주세요.



- 해당 폴더에 있을  VSCode-darwin-universal.zip 을 더블 클릭해서 압축을 풀어주세요.

VS Code 설치 (3/4)

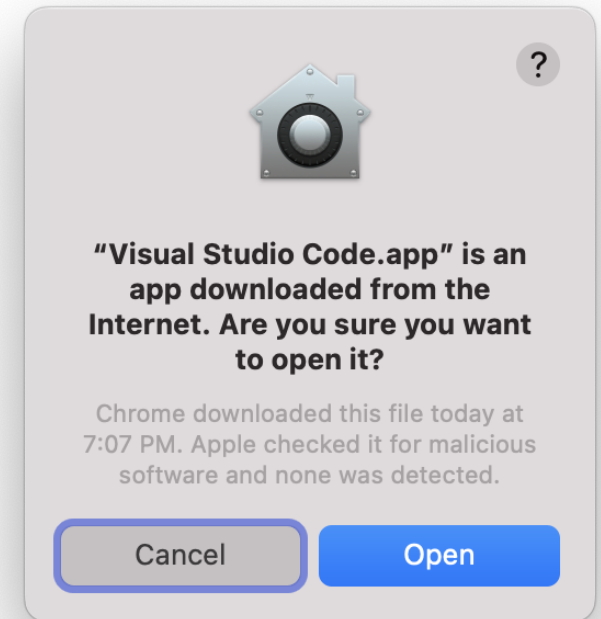
- 압축을 해제하면 그 폴더에 Visual Studio Code.app 이라는 이름의 파일이 생성될 거예요.
- (중요)그 파일을 그대로, Finder 창 왼쪽 목록에 있는 Applications(또는 응용 프로그램)으로 드래그 & 드롭해서 위치를 옮겨주세요.



- 이동한 폴더를 들어가 Visual Studio Code.app을 더블 클릭하여 실행해 주세요.

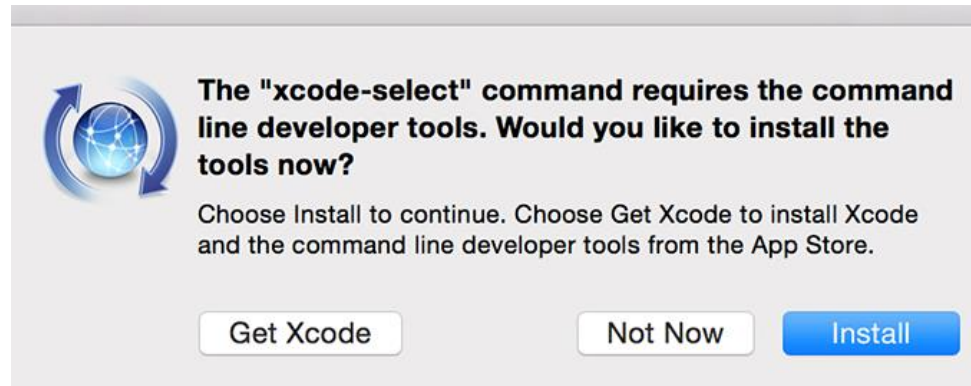
VS Code 설치 (4/4)

- 인터넷에서 다운 받은 파일을 실행(설치)해도 괜찮은 지 묻는 창이 뜨면 Open(또는 열기)를 눌러 주세요.
 - 누르면 설치가 끝나요
- 자주 사용하는 VS Code는 화면 아래 앱 목록에 고정해 두고 사용하면 좋을듯



gcc 설치하기 (1/8)

- Command + Space 키를 눌러 Terminal 앱을 검색해서 실행해주세요.
- Terminal 앱에서 `xcode-select --install` 을 입력해 주세요.



- 위와 같은 창이 뜨면
Install(또는 설치)를 눌러 기본 명령어 도구를 설치해 주세요.

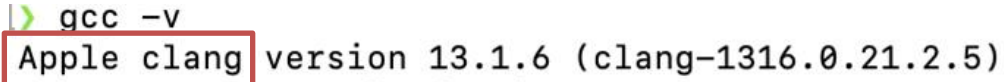
gcc 설치하기 (2/8)

- 설치가 끝나면 이번에는 Terminal 앱에서 `gcc -v` 를 입력해 주세요.

- 입력했을 때
오른쪽 스샷과 비슷한 내용이 출력된다면
정상적으로 설치된 것이에요:

```
l> gcc -v
Apple clang version 13.1.6 (clang-1316.0.21.2.5)
Target: arm64-apple-darwin21.5.0
Thread model: posix
InstalledDir: /Applications/Xcode.app/Contents/Develop
.xctoolchain/usr/bin
```

gcc 설치하기 (3/8)

- ...하지만 방금 설치된 것은
우리 수업에서 다룰 기능들 중 일부를 아직 지원하지 않는,
Apple에서 만든 clang 컴파일러예요.
 - 원래 이름은 clang인데 gcc인 척 하고 있어요 
- 그러니 목표 달성을 위해 몇 가지 작업을 더 진행해 봅시다:
 - Brew 패키지 관리자를 사용하여 '진짜 gcc'를 설치
 - Apple clang 대신 진짜 gcc를 쓸 수 있도록 설정 변경
 - 이걸 조금 어려운데 미리 자동화시켜 놓았어요.
수업 페이지에 있는 change_compiler.sh.zip을 다운로드해 주세요

gcc 설치하기 (4/8)

- Brew 패키지 관리자 설치
 - Terminal 앱에서 다음 명령어를 입력해 주세요:
 - `/bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"`
 - 길어서 두 줄에 걸쳐 적어 놓았는데, 여러분이 실제로 적을 땐 한 줄에 다 적어야 해요 (그냥 복사해요)
 - 엔터 치면 brew 패키지 관리자를 설치하겠냐는 설치 창이 보일 거예요. 엔터 키를 한번 더 눌러 설치를 진행할 수 있어요

gcc 설치하기 (5/8)

- Brew 패키지 관리자로 진짜 gcc 설치
 - Terminal 앱에서 다음 명령어를 입력해 주세요:
 - `brew install gcc@12`
 - 대충 아래와 같이 출력되면 설치가 잘 되고 있는 셈이에요:

```
==> Downloading https://ghcr.io/v2/homebrew/core/gcc/12/manifests/12.1.0_1
Already downloaded: /Users/hgs/Library/Caches/Homebrew/downloads/4f32b9f47ac3328f3cc3edc1c7b38244b7b03fc16267f9307f5fd9e
b5eda6911--gcc@12-12.1.0_1.bottle_manifest.json
==> Downloading https://ghcr.io/v2/homebrew/core/gcc/12/blobs/sha256:19561d68c1bf245edd39d1c569801f8f4d5cc1729403f18df4a
Already downloaded: /Users/hgs/Library/Caches/Homebrew/downloads/6bebb8e3bc549bb9386ec0707363599391ed0b55bdd7d9c439f276d
25a82dd59--gcc@12--12.1.0_1.arm64_monterey.bottle.tar.gz
==> Reinstalling gcc@12
==> Pouring gcc@12--12.1.0_1.arm64_monterey.bottle.tar.gz
🍺 /opt/homebrew/Cellar/gcc@12/12.1.0_1: 1,422 files, 261.6MB
==> Running `brew cleanup gcc@12`...
```

gcc 설치하기 (6/8)

- Brew 패키지 관리자로 진짜 gcc 설치
 - 하지만 진짜 gcc를 설치해도 Terminal 앱에서 `gcc -v` 엔터 치면... 여전히 아까와 동일한 Apple clang이 등장하는 것을 알 수 있어요:

```
> gcc -v
Apple clang version 13.1.6 (clang-1316.0.21.2.5)
```

- 이것은 방금 설치한 진짜 gcc가 'gcc'가 아닌 'gcc-12'라는 이름으로 설치되었기 때문이에요.
 - Terminal 앱에서 `gcc-12 -v` 엔터 치면 아래와 같이 GCC 버전 12가 설치된 것을 확인할 수 있어요:

```
> gcc-12 -v
Using built-in specs.
COLLECT_GCC=gcc-12
COLLECT_LTO_WRAPPER=/opt/homebrew/Cellar/gcc@12/12.1.0_1/bin/./libexec/gcc/aarch64-apple-darwin21/12/lto-wrapper
Target: aarch64-apple-darwin21
```

gcc 설치하기 (7/8)

- 진짜 gcc를 쓸 수 있도록 설정 변경
 - 그래서 이제는 이름 바꾸기를 진행해야 해요
- 수업 페이지에 있는 change_compiler.sh.zip을 다운로드한 다음 압축을 풀어 change_compiler.sh를 준비해 주세요

 **(Mac용) change_compiler.sh.zip**



change_compiler.sh.zip

gcc 설치하기 (8/8)

- 진짜 gcc를 쓸 수 있도록 설정 변경
 - Terminal 앱에서 change_compiler.sh가 있는 폴더로 디렉터리를 변경한 후, `./change_compiler.sh` 를 입력해서 그 파일의 내용을 실행해 주세요
 - 중요한 내용을 변경하기 때문에 암호를 물어볼 수 있어요
Macbook 로그인할 때 쓰는 암호를 치면 돼요(몇 글자 쳤는지 안 보여주니 외워서 쳐야 해요)

```
[> ./change_compiler.sh  
Password: [REDACTED]
```

- Changed to gcc successfully 라고 출력되면 성공이에요.
이제 Terminal에서 `gcc -v` 엔터 치면 진짜 gcc가 나올 거예요

```
[> ./change_compiler.sh  
[Password:  
Changed to gcc successfully  
[> gcc -v  
Using built-in specs.  
COLLECT_GCC=gcc  
COLLECT_LTO_WRAPPER=/opt/homebrew/Cellar/gcc@12/12.1.0_1/bin/../../libexec/gcc/aarch64-apple-darwin21/12/lto-wrapper  
Target: aarch64-apple-darwin21
```

프로그램 만드는 방법 소개 (1/11)

- 이제 수업 페이지에 있는 env_test.cpp를 다운로드한 다음
아까 설치했던 VS Code를 통해 열어볼게요.
 - 이 파일 안에 있는 내용은 우리 수업 극후반에 등장할 예정이니
내용은 안 봐도 괜찮아요



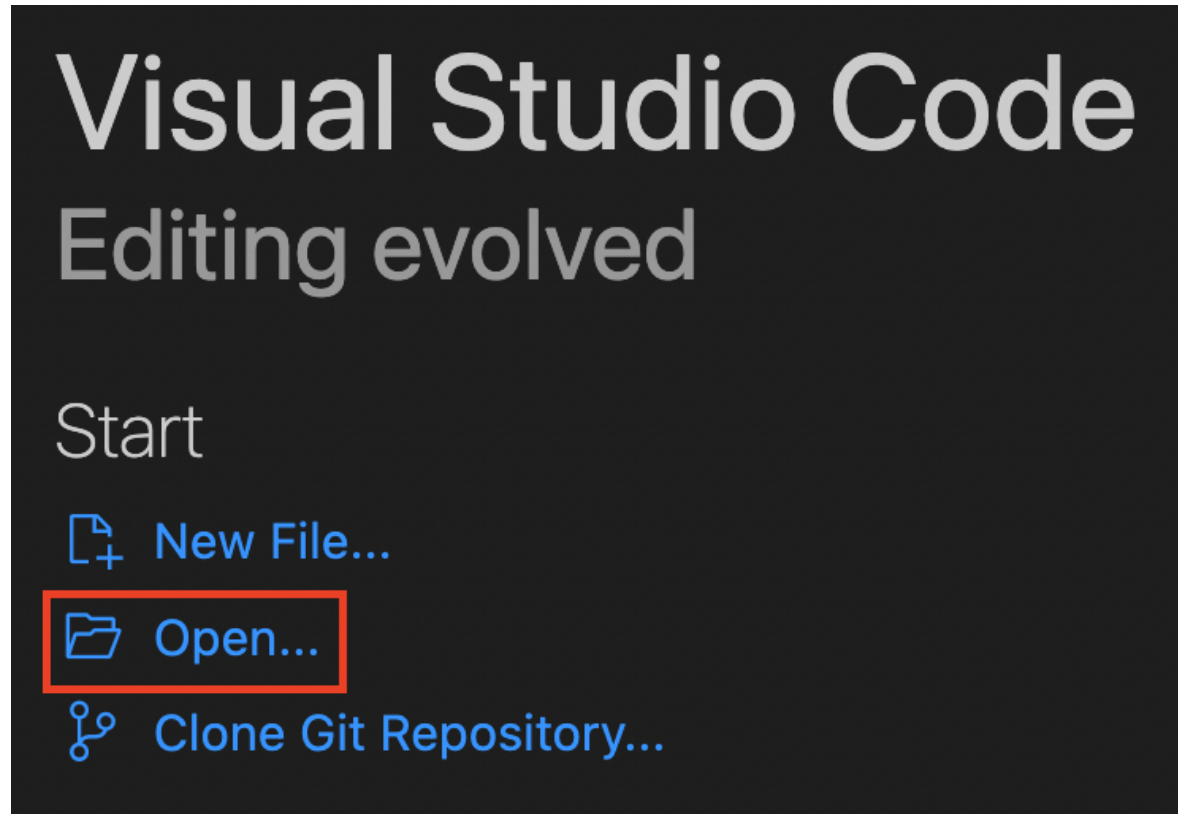
(NPC용) env_test.cpp



파일

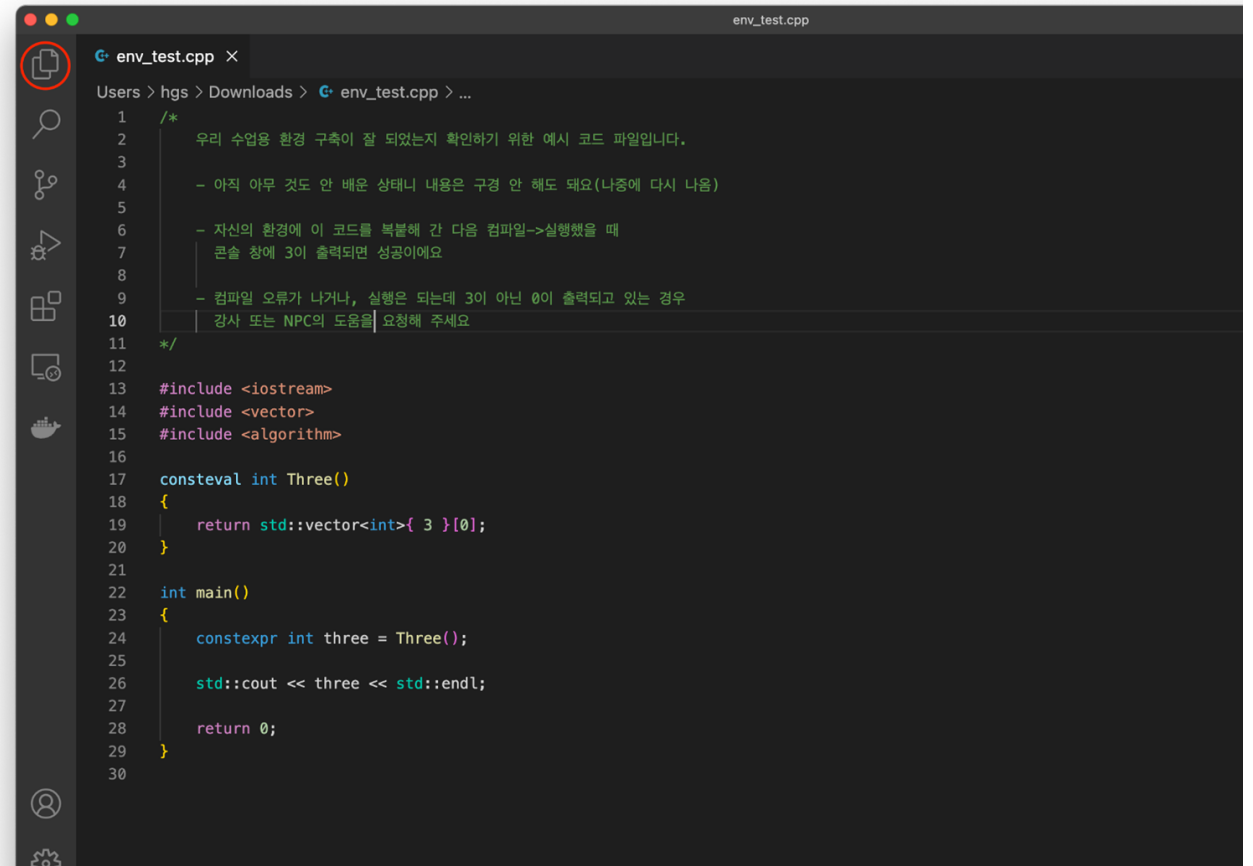
프로그램 만드는 방법 소개 (2/11)

- VS Code를 실행시켜 env_test.cpp 파일을 열어주세요.



프로그램 만드는 방법 소개 (3/11)

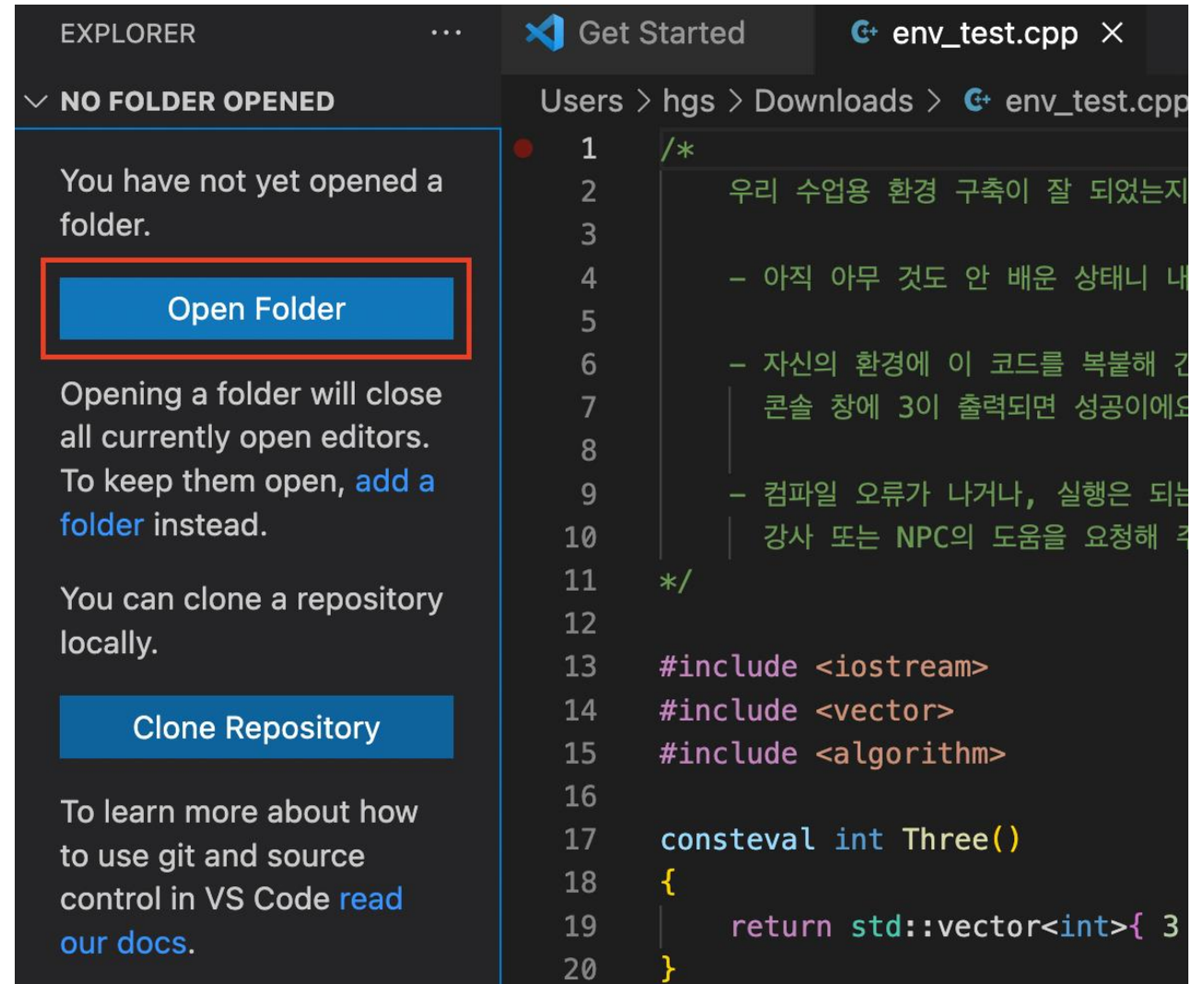
- 왼쪽에 있는 폴더 보기 버튼을 눌러
파일이 들어있는 폴더를 확인해주세요.
- 지금은
코드에서 빨간 줄이 보여도
괜찮아요



```
env_test.cpp X
Users > hgs > Downloads > env_test.cpp > ...
1  /*
2     우리 수업용 환경 구축이 잘 되었는지 확인하기 위한 예시 코드 파일입니다.
3
4     - 아직 아무 것도 안 배운 상태니 내용은 구경 안 해도 돼요(나중에 다시 나옴)
5
6     - 자신의 환경에 이 코드를 복붙해 간 다음 컴파일->실행했을 때
7       콘솔 창에 3이 출력되면 성공이에요
8
9     - 컴파일 오류가 나거나, 실행은 되는데 3이 아닌 0이 출력되고 있는 경우
10      감사 또는 NPC의 도움을 요청해 주세요
11  */
12
13  #include <iostream>
14  #include <vector>
15  #include <algorithm>
16
17  constexpr int Three()
18  {
19      return std::vector<int>{ 3 }[0];
20  }
21
22  int main()
23  {
24      constexpr int three = Three();
25
26      std::cout << three << std::endl;
27
28      return 0;
29  }
30
```

프로그램 만드는 방법 소개 (4/11)

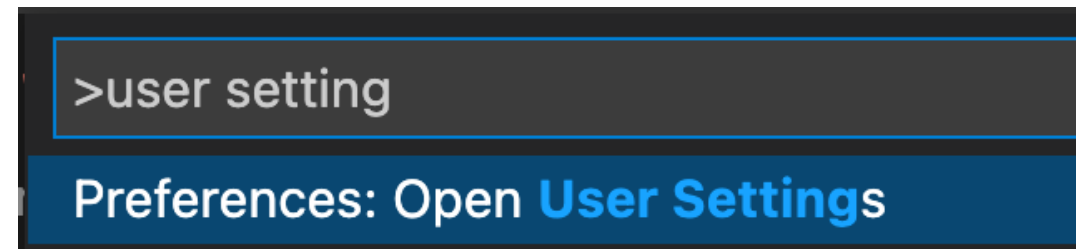
- 폴더 열기(Open Folder)를 눌러 현재 파일이 포함된 폴더를 열어주세요.



프로그램 만드는 방법 소개 (5/11)

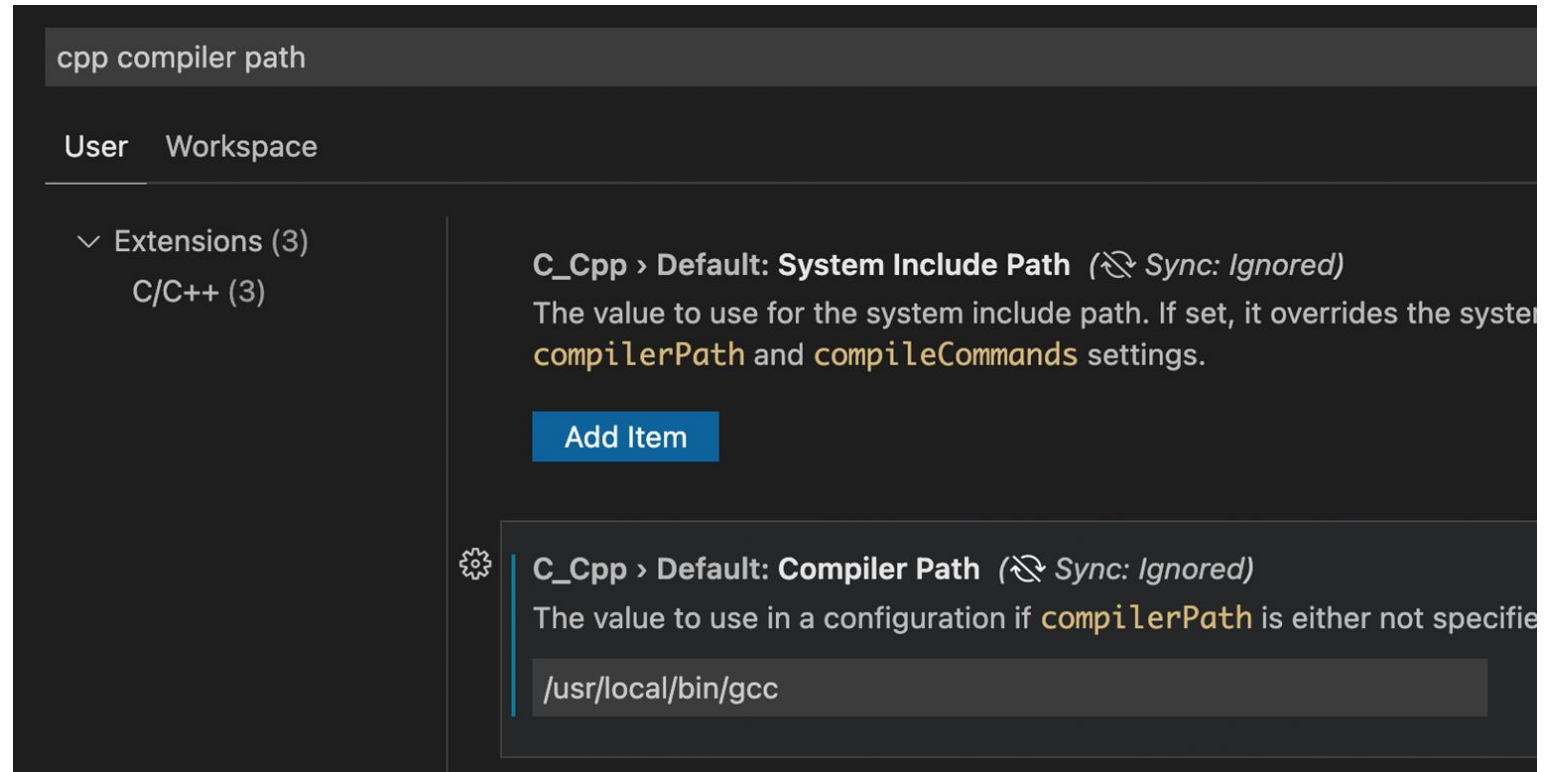
- 이 상태에서 Command + Shift + P 를 눌러 user settings를 찾아주세요.

```
1  /*
2     우리 수업용 환경 구축이 잘 되었는지 확인하기 위한 예시 코드 파일입니다.
3
4     - 아직 아무 것도 안 배운 상태니 내용은 구경 안 해도 돼요(나중에 다시 나옴)
5
6     - 자신의 환경에 이 코드를 복붙해 간 다음 컴파일->실행했을 때
7       콘솔 창에 3이 출력되면 성공이에요
8
9     - 컴파일 오류가 나거나, 실행은 되는데 3이 아닌 0이 출력되고 있는 경우
10      감사 또는 NPC의 도움을 요청해 주세요
11  */
12
13  #include <iostream>
14  #include <vector>
15  #include <algorithm>
16
17  constexpr int Three()
18  {
19      return std::vector<int>{ 3 }[0];
20  }
21
22  int main()
23  {
24      constexpr int three = Three();
25
26      std::cout << three << std::endl;
27
28      return 0;
29  }
30
```



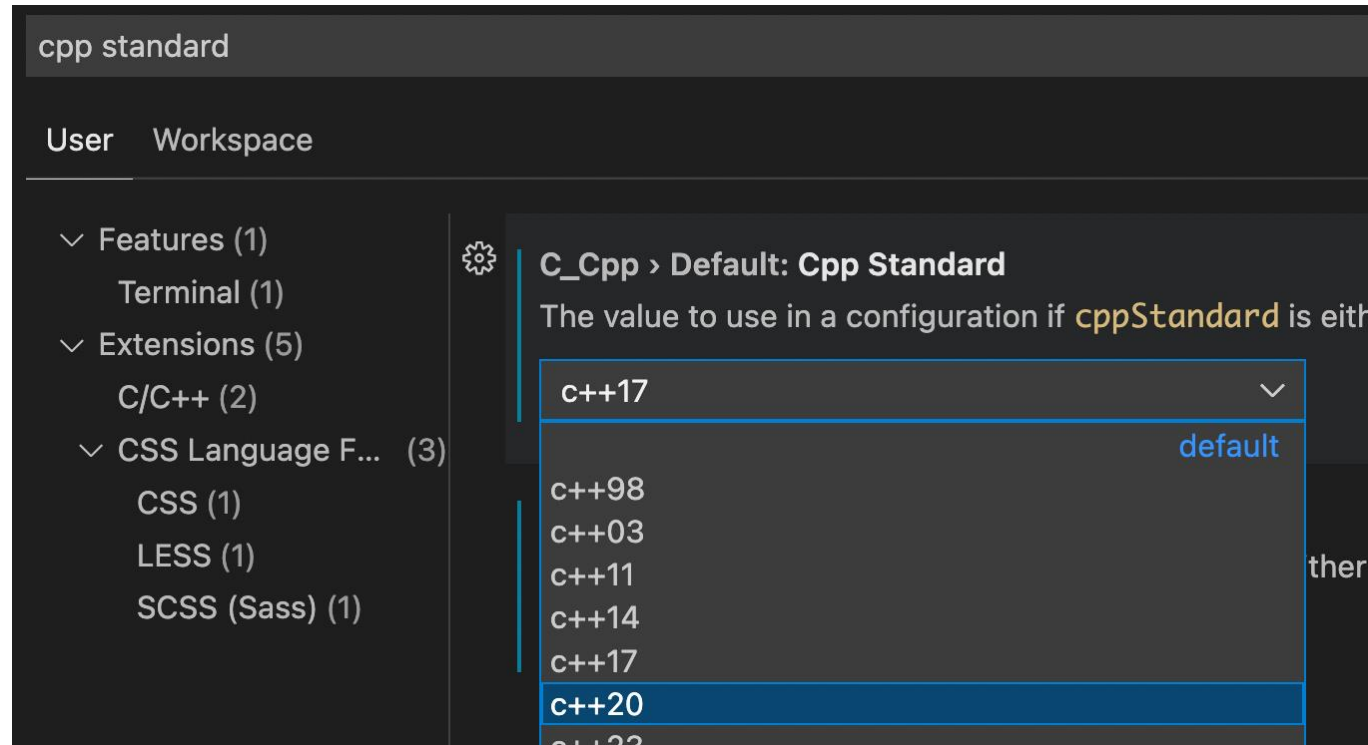
프로그램 만드는 방법 소개 (6/11)

- 설정 창의 검색 칸에 `cpp compiler path` 를 입력해서
C_Cpp > Default: Compiler Path 의 값을
`/usr/local/bin/gcc` 로 설정해주세요.
- (만약 해당 속성이 나오지 않는다면, 다음 [링크](#)에서 extension을 install을 눌러 설치해 주세요)



프로그램 만드는 방법 소개 (7/11)

- 다시, 이번에는 cpp standard 를 검색해서
C_Cpp > Default: Cpp Standard의 값을 c++20으로 설정해주세요.



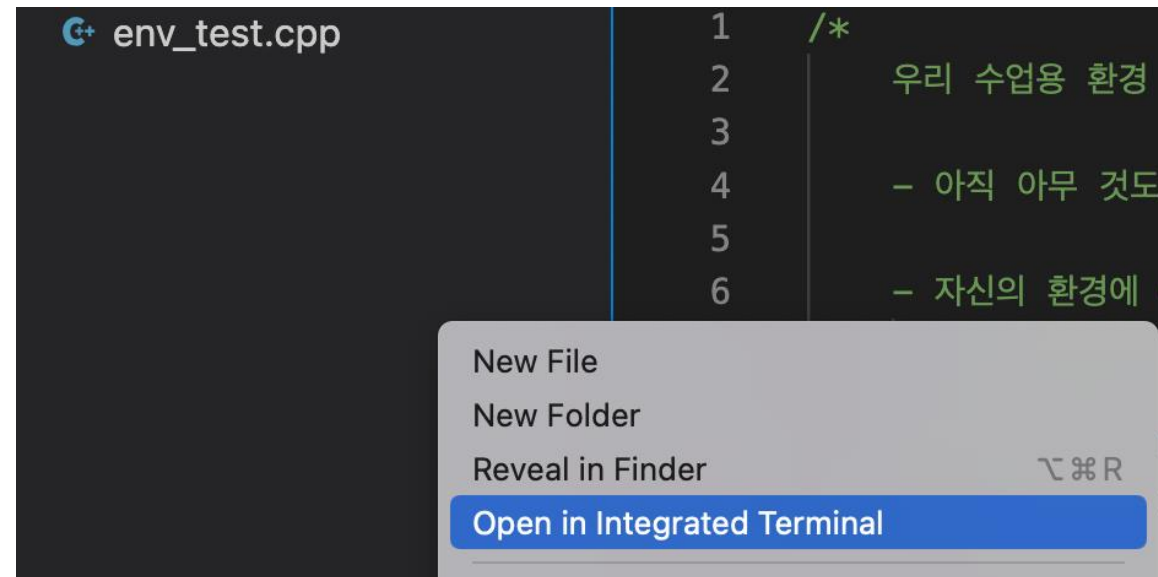
프로그램 만드는 방법 소개 (8/11)

- 설정 창을 닫고 다시 env_test.cpp을 열면,
아까와 달리
빨간 줄이 사라진 것을 확인할 수 있어요.

```
env_test.cpp x
env_test.cpp > ...
1  /*
2     우리 수업용 환경 구축이 잘 되었는지 확인하기 위한 예시 코드 파일입니다.
3
4     - 아직 아무 것도 안 배운 상태니 내용은 구경 안 해도 돼요 (나중에 다시 나옴)
5
6     - 자신의 환경에 이 코드를 복붙해 간 다음 컴파일->실행했을 때
7       콘솔 창에 3이 출력되면 성공이에요
8
9     - 컴파일 오류가 나거나, 실행은 되는데 3이 아닌 0이 출력되고 있는 경우
10      강사 또는 NPC의 도움을 요청해 주세요
11 */
12
13 #include <iostream>
14 #include <vector>
15 #include <algorithm>
16
17 constexpr int Three()
18 {
19     return std::vector<int>{ 3 }[0];
20 }
21
22 int main()
23 {
24     constexpr int three = Three();
25
26     std::cout << three << std::endl;
27
28     return 0;
29 }
30
```

프로그램 만드는 방법 소개 (9/11)

- 이제 gcc 관련 설정은 모두 끝났으니,
내가 작성한 코드로 프로그램 만드는 방법을 확인해 볼게요.
- 좌측 메뉴에 마우스 포인터를 가져다 대고 우클릭 →
내장 터미널에서 열기(Open in Integrated Terminal)을 누르면...



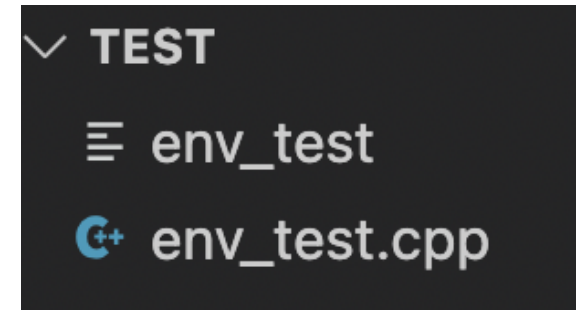
프로그램 만드는 방법 소개 (10/11)

- 다음과 같이 내장 터미널이 열리면서 명령어를 입력할 수 있게 돼요:



The screenshot shows an IDE interface. At the top, a code editor displays two lines of C++ code: line 24 has `constexpr int three = Three();` and line 25 is empty. Below the code editor, there are four tabs: PROBLEMS, OUTPUT, DEBUG CONSOLE, and TERMINAL. The TERMINAL tab is selected and active, showing a command prompt `~/Downloads/test >` with a cursor.

- 내장 터미널에서
`gcc -std=c++20 env_test.cpp -o env_test`
...를 명령어로 입력하게 되면,
`gcc`를 사용하여 `env_test.cpp`를
`C++20` 표준에 입각해 다룸으로써
최종적으로 `env_test` 라는 이름의 파일을 만들게 돼요.



프로그램 만드는 방법 소개 (11/11)

- 이제 내장 터미널에서 `./env_test` 엔터 치면 바로 실행해 볼 수 있어요.
 - 3이 출력되면 정상이에요.
VS Code 설정이 아직 덜 되었다면
0이 출력되거나, 아예 빨간 줄이 안 없어지거나 했을 거예요

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
> g++ -std=c++20 env_test.cpp -o env_test
> ./env_test
3
```

부록: VS Code 소개 (1/2)

- Visual Studio Code는 Microsoft에서 만든 무료, 오픈-소스 파일 편집기예요.
- 우리 수업 표준 환경으로 쓰이는 Visual Studio는 IDE(Integrated Development Environment)의 일종이에요.
 - IDE는 파일 편집기 + 컴파일러 + ... 가 포함된 풀패키지라고 보면 돼요
 - 그런데 아쉽게도 Mac에서는 Visual Studio를 C/C++용으로 사용할 수 없어요
- 그래서 우리 수업에서 Mac을 쓸 때는, 각종 extension이 설치된 VS Code를 통해 소스 코드를 작성하고, 프로그램 만들 때는 gcc를 직접 사용하는 방식으로 진행할 예정이에요.



부록: VS Code 소개 (2/2)

- 자주 사용하는 단축키
 - 주석 설정/해제: Command + /
 - 선택 영역 코드 포매팅: Command + K, F
 - Command 키를 누른 채로 K 눌렀다가 떼고 F 눌러요
 - 코드 내에서 찾기: Command + F
 - 전체 폴더에서 찾기: Command + Shift + F
 - F12: 커서가 가리키는 이름에 대한 정의가 적혀 있는 곳으로 이동



마무리

- (이 슬라이드는 강사가 적었어요)
 - 여러분과 함께 의미 있는 시간을 보내기 위해 NPC 친구들이 와 있어요
 - 사실 NPC 친구들도 (대학원생이 아니라) 알바, 다른 계절수업, 봉사활동 등등을 병행중인, 여러분과 동등한 학부생 친구들이에요
- 강사와 NPC 친구들이 (본인들 안 바쁠 때) 최대한 도와드릴 예정이니, 학교나 집에서, 여러분이 하고 싶은 만큼 마음껏 공부하고 고민해 보고 갔으면 좋겠어요