

C++ 실습 1

C++ 실습 준비

- 개발환경 구축을 위하여 다음의 프로그램 설치가 필요합니다.
 - Compiler - MinGW or g++
 - IDE - VSCode
- 이미 자신이 익숙한 개발환경이 있다면 굳이 새로 환경을 구축할 필요는 없습니다.
- 다운로드 링크 등은 Course GitHub에서도 찾을 수 있습니다.
- 이후의 개발환경 설정 실습내용은 슬라이드의 순서를 지키며 따라하는 것이 예기치 못한 오류 방지를 위해 권장됩니다.

Compiler

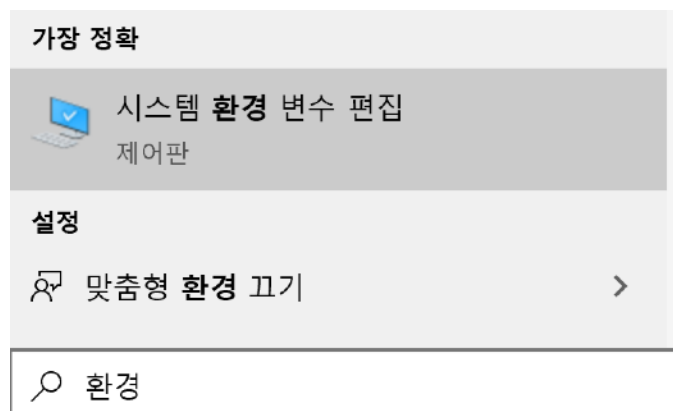
- 컴파일러(Compiler): 자신이 작성한 C++ 코드를 컴퓨터가 이해할 수 있도록 변환해주는 역할을 함.
- OS Dependent: 각 운영체제(Operating System, OS)별로 다른 컴파일러가 필요.
 - Windows: MinGW
 - macOS: g++ → Xcode 설치하면 자동으로 따라옴.
 - Linux: g++ → 보통 설치되어 있으나 없는 경우 g++ 패키지 설치.
 - CentOS: `sudo yum install gcc-c++`
 - Ubuntu: `sudo apt install g++`

Windows Setup

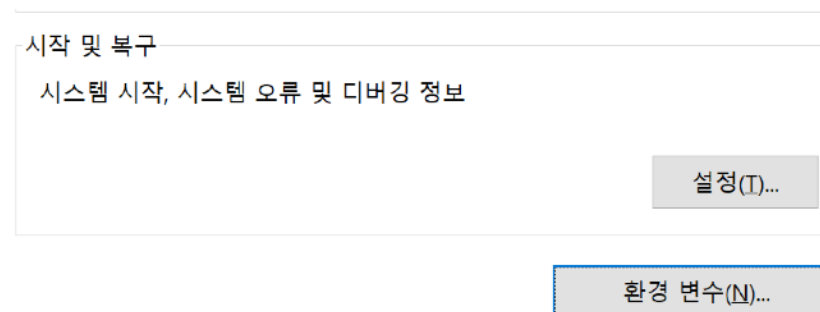
- Windows에서 MinGW 설치 프로그램을 받은 경우.
 - 설치 프로그램 실행 후 Package설명을 보고 Basic MinGW와 C++ Compiler로 나온 것들을 체크한 뒤, Installation → Apply Change를 선택하면 설치가 진행됩니다.
- 설치가 완료된 후에는 PATH 설정을 해줘야 합니다.

PATH Setup

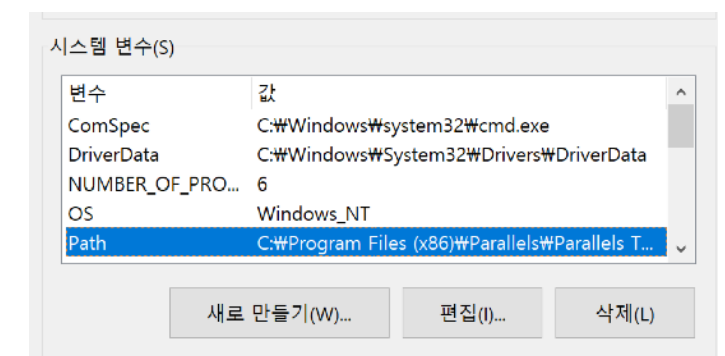
1. Windows + S를 눌러 검색창에서 '환경' 입력



2. 시스템 환경 변수 편집 창에서 '환경 변수' 버튼을 찾아 클릭

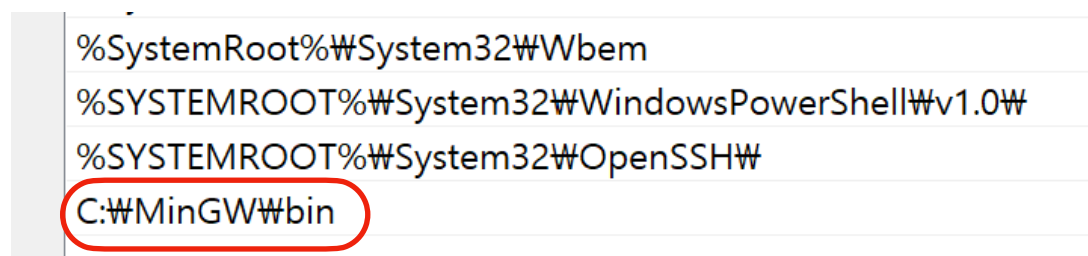


3. 시스템 변수에서 Path를 찾아 선택 → '편집' 클릭



*공용 컴퓨터의 경우 시스템 변수가 아닌
특정 사용자 변수를 편집해야 할 수 있음.

4. 나타난 화면에서 '새로 만들기' 를 누른 후 다음의 경로를 입력



* 표시된 것이 기본 경로이지만, 만약 MinGW를 다른 경로에
설치한 경우 g++.exe. 파일의 위치를 찾아 그 경로를 입력

****설정이 잘 되었는지 명령 프롬프트(검색에서 'cmd')를
실행한 후, 'g++ -v' 명령어를 입력하여 컴파일러 정보가
표시되는지 확인**

Integrated Development Environment

- IDE: 통합 개발환경으로 소스코드 작성에서부터 테스트, 빌드, 디버깅까지 지원하는 프로그램. (e.g., VSCode, Eclipse, IntelliJ, PyCharm, etc.)
- 주요 기능들
 - Syntax Highlight
 - Auto Completion
 - Build
 - Program Execution
 - Debugging Support

Syntax Highlight

- 다른 색상으로 문법적으로 다른 위치의 단어들을 표시해 주는 기능.
- 코드의 가독성이 훨씬 좋아져 생산성을 높여줌.
- 문법적인 오류가 있는 경우 밑줄로 오류를 표시해주는 기능이 제공되는 경우도 있음.

```
#include<iostream>
using namespace std;

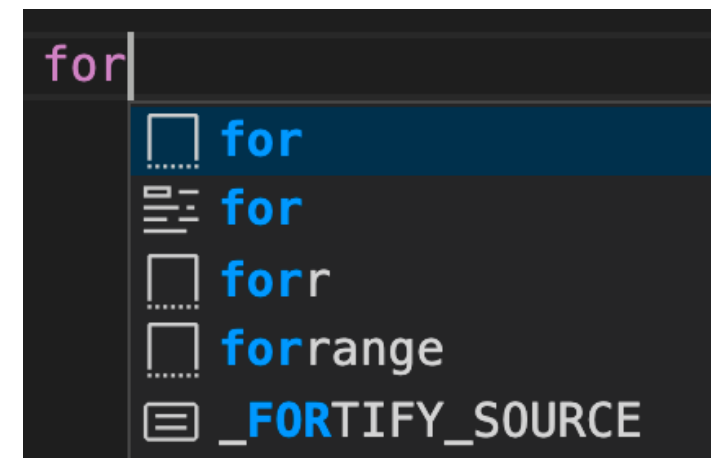
int main() {
    cout << "Hello World!\n";
    return 0;
}
```

```
#include<iostream>
using namespace std;

int main() {
    cout << "Hello World!\n";
    return 0;
}
```

Auto Completion

- 코드의 일부만 입력해주면 자동으로 나머지를 완성해 주는 기능.
- 개발자의 생산성 향상에 막대한 영향을 끼치는 기능임.
- 이를 효율적으로 하기위한 다양한 연구들이 다수 존재함.



```
for (size_t i = 0; i < count; i++)  
{  
    /* code */  
}
```


Build

- 자동으로 필요한 파일들을 컴파일하여 실행가능한 프로그램을 만들어주는 기능.
- 대형 프로그램에서 다수의 파일이 존재하는 경우 컴파일 작업 자체가 매우 복잡하게 됨.
- IDE에서는 명령어 하나로 변경된 파일들을 다시 컴파일하여 수정 사항이 반영된 실행 프로그램을 생성할 수 있음.

Program Execution

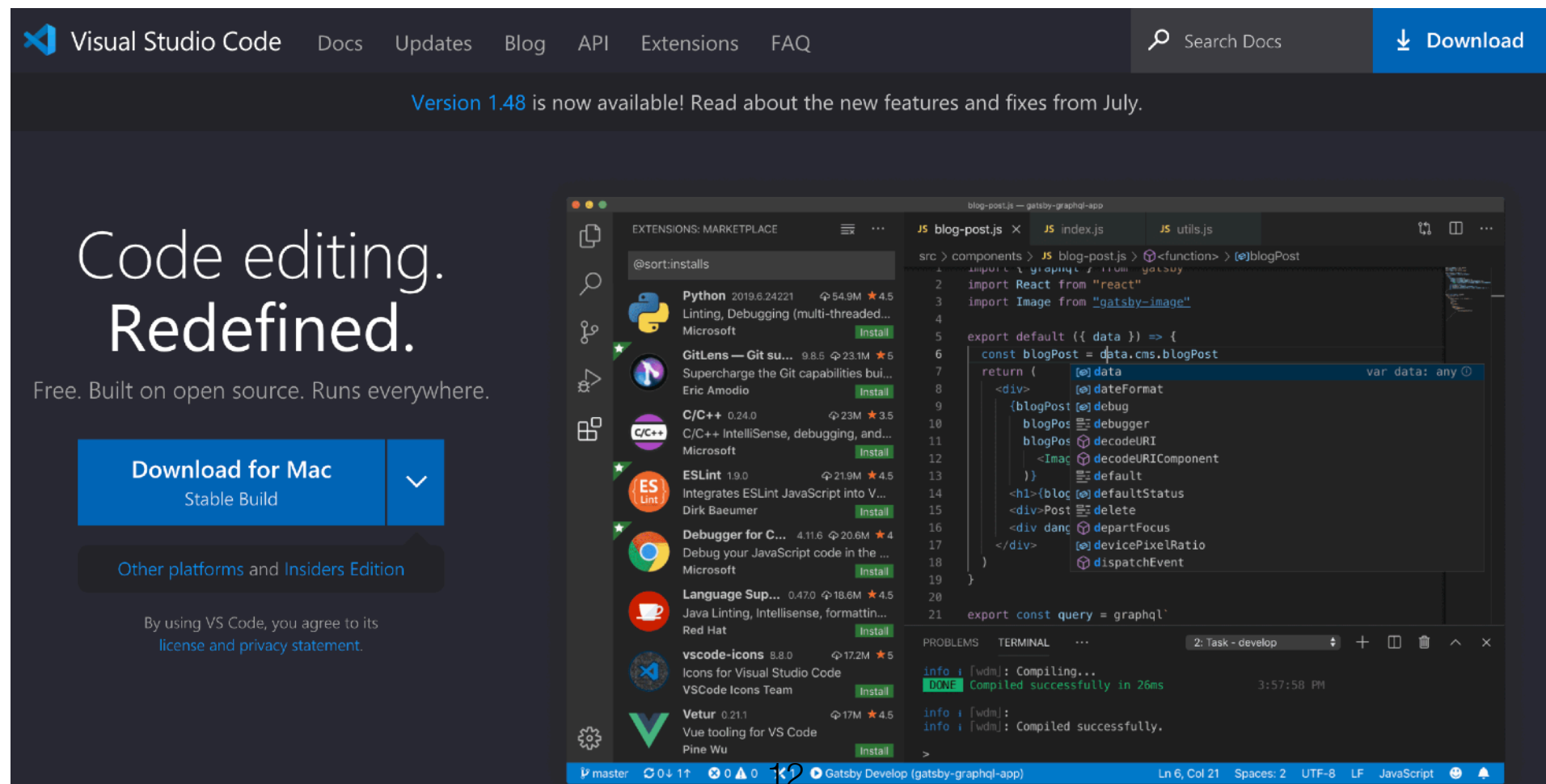
- IDE 내부에서 바로 프로그램을 실행하여 결과를 확인할 수 있음.
- 코드를 변경한 후 빌드 + 실행을 통해 바로 변경사항이 반영된 프로그램을 확인하며 개발을 진행하는 것이 가능.
- 프로그램에 입력이 필요하거나 특정한 실행 환경이 필요한 경우 한 번의 설정을 통해 반복적으로 실행할 수 있다는 장점이 있음.

Debugging Support

- 언어별로 차이는 있으나, 일반적으로 코드를 순차적으로 수행하며 실행상태를 확인하는 것이 가능.
- 예를 들어, 코드의 15번째 줄에 Break Point를 설정하고 디버깅 모드로 실행하면 프로그램이 그 곳에 도달한 시점에서 일시 멈춤.
- 그 상태에서 각각의 변수들이 어떤 값을 가지고 있는지 확인하여 현재 프로그램의 상태가 의도한 대로인지 확인 가능.
- 또한 멈춘 프로그램을 다시 정해진 만큼씩 실행하며 상태가 어떻게 변해가는지 관찰하는 기능이 지원되기도 함.

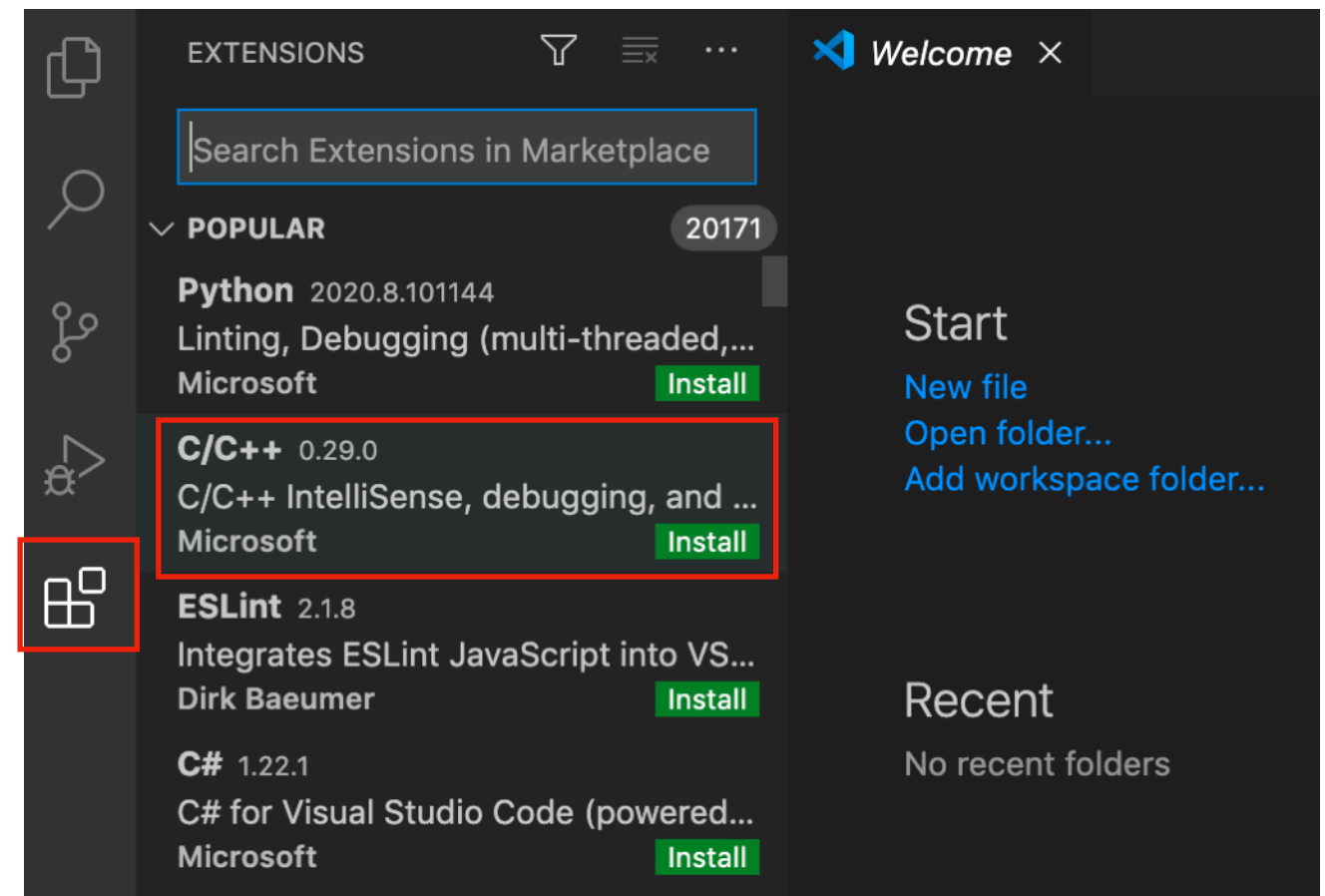
VSCode

- Visual Studio Code: Microsoft에서 개발한 무료 IDE.
- Windows, Mac, Linux를 모두 지원
- Extension으로 다양한 언어를 지원



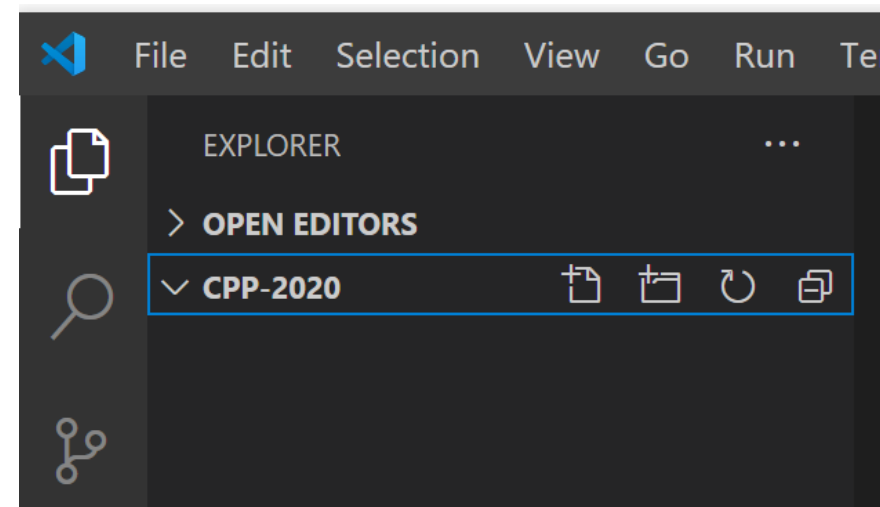
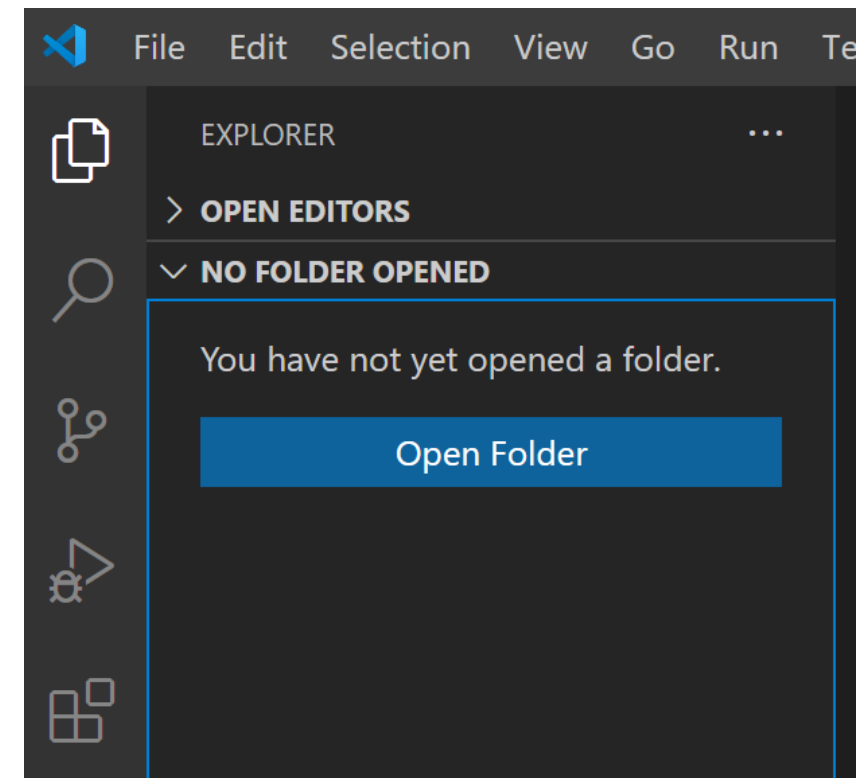
Installing Extensions

- 화면 왼쪽 끝의 사각형 모양을 눌러 EXTENSIONS 메뉴를 엽니다.
- C/C++ Extension이 일반적으로 상단에 나타나고, 없는 경우는 검색하여 찾습니다.
- Install 버튼을 눌러 설치합니다.

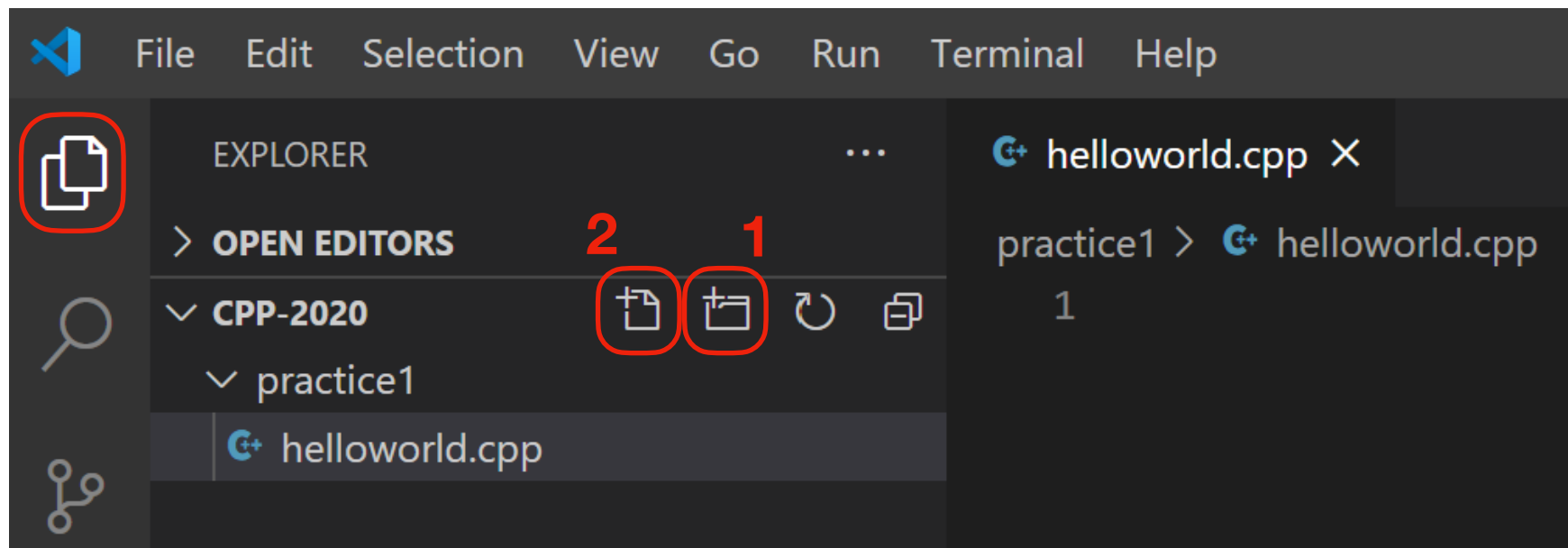


Workspace

- 개발을 위한 작업공간.
- 여러분의 프로그램에 관련된 파일들이 모여있는 집합소.
- Workspace별로 환경설정을 다르게 하거나, workspace 내에서 여러 프로그램들이 하나의 설정을 공유하는 것도 가능.
- VSCode에서는 File > Open으로 폴더(e.g., cpp-2020)를 생성하고 여는 것으로 workspace 생성 및 열기가 가능함.



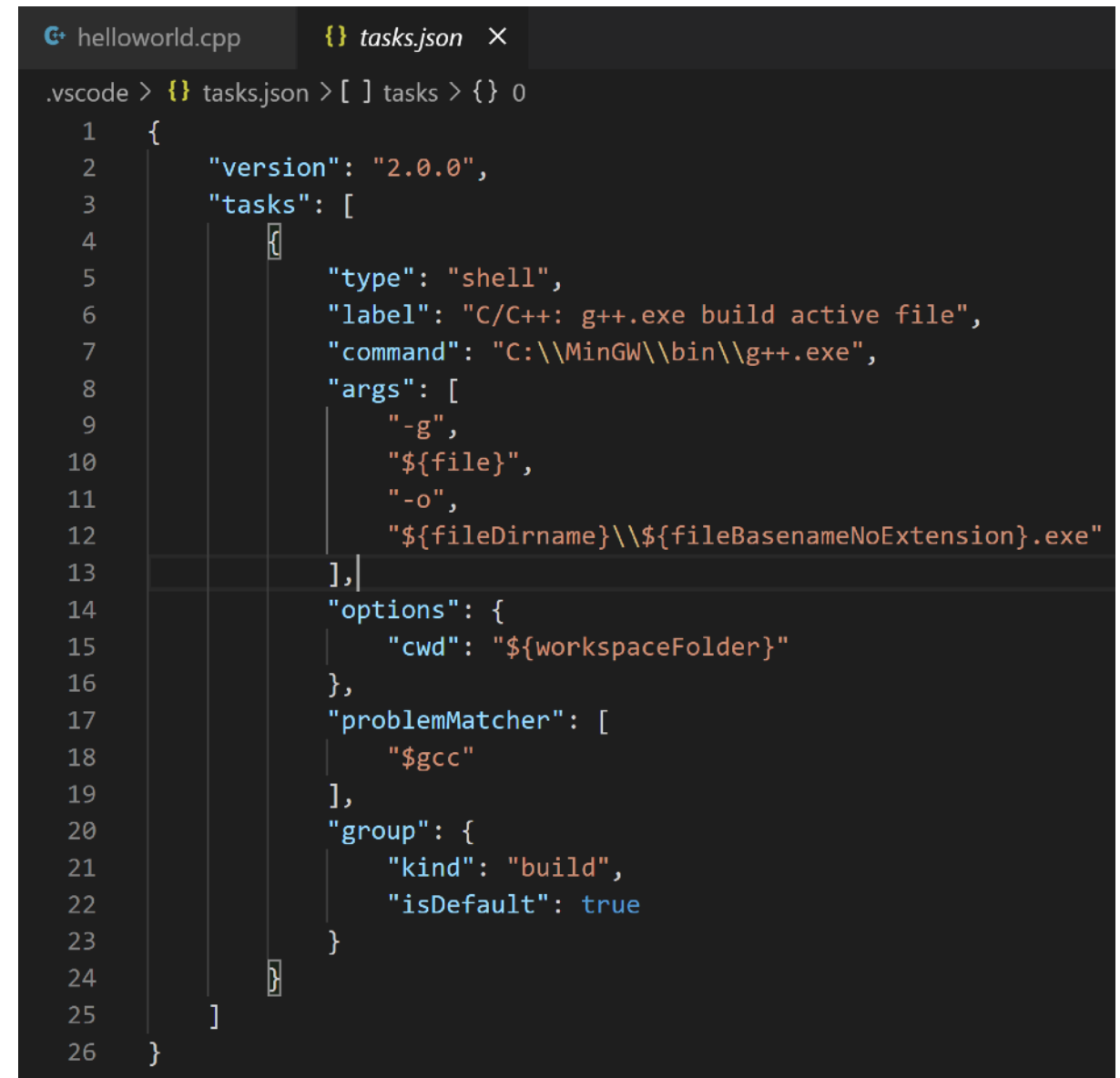
첫 C++ 파일 생성



- EXPLORER에서 CPP-2020 옆의 두번째 아이콘(1)을 클릭하여 practice1 폴더를 생성.
- 첫번째 아이콘(2)을 클릭하여 폴더 아래에 새 파일 helloworld.cpp를 생성.
- 자동으로 helloworld.cpp가 오른쪽에 열림.
- 만약 아직 C/C++ Extension을 설치하지 않은 상태라면 설치하라는 안내가 나오므로 설치하면 됩니다.

빌드 및 실행 설정

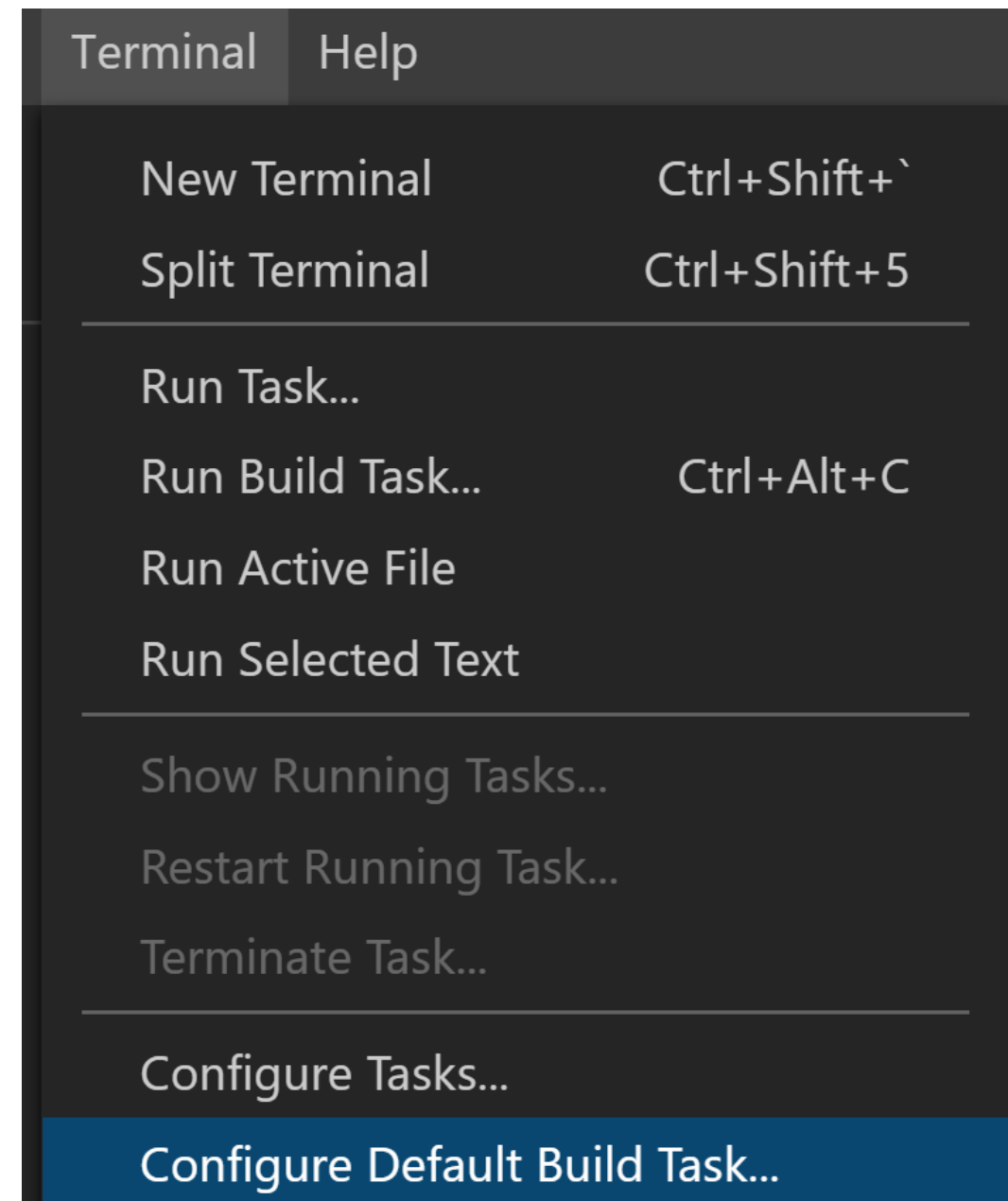
- 앞으로 작성할 코드를 빌드(컴파일)하고 실행하기 위하여 추가적인 설정이 필요함.
- VSCode에서는 이 때 실행되어야 할 Task를 tasks.json 파일로 저장함.
- 오른쪽은 Windows에서 VSCode가 기본 제공하는 tasks.json의 예시임.
- 구글에서 'VSCode C++ 개발환경 구축'을 검색하면 다양한 설정을 확인할 수 있음.



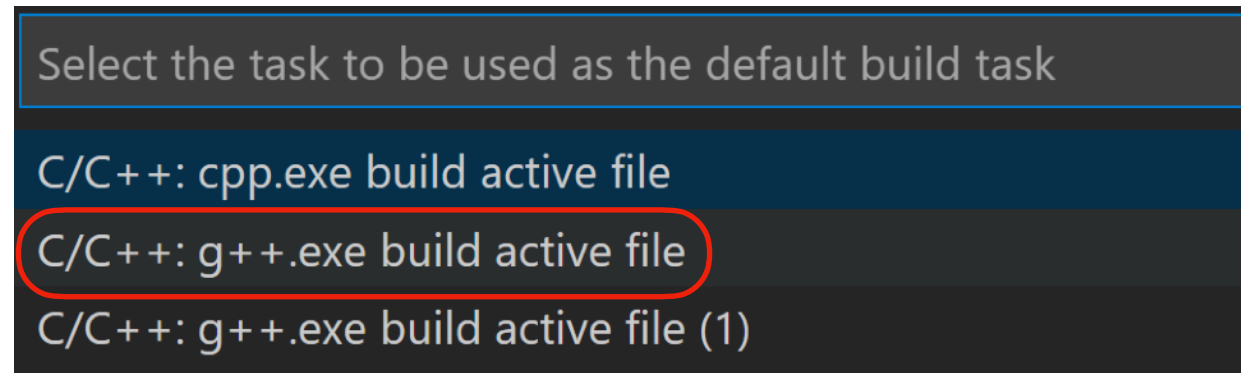
```
.vscode > {} tasks.json > [ ] tasks > {} 0
1  {
2      "version": "2.0.0",
3      "tasks": [
4          {
5              "type": "shell",
6              "label": "C/C++: g++.exe build active file",
7              "command": "C:\\MinGW\\bin\\g++.exe",
8              "args": [
9                  "-g",
10                 "${file}",
11                 "-o",
12                 "${fileDirname}\\${fileBasenameNoExtension}.exe"
13             ],
14              "options": {
15                  "cwd": "${workspaceFolder}"
16              },
17              "problemMatcher": [
18                  "$gcc"
19              ],
20              "group": {
21                  "kind": "build",
22                  "isDefault": true
23              }
24          }
25      ]
26  }
```


빌드 및 실행 설정(1)

- 설정을 위해 우선 아까 생성한 helloworld.cpp를 선택.
- 이 후 Terminal → Configure Default Build Task를 선택.
- 자동으로 C++와 관련된 빌드 설정들의 리스트가 나타남.



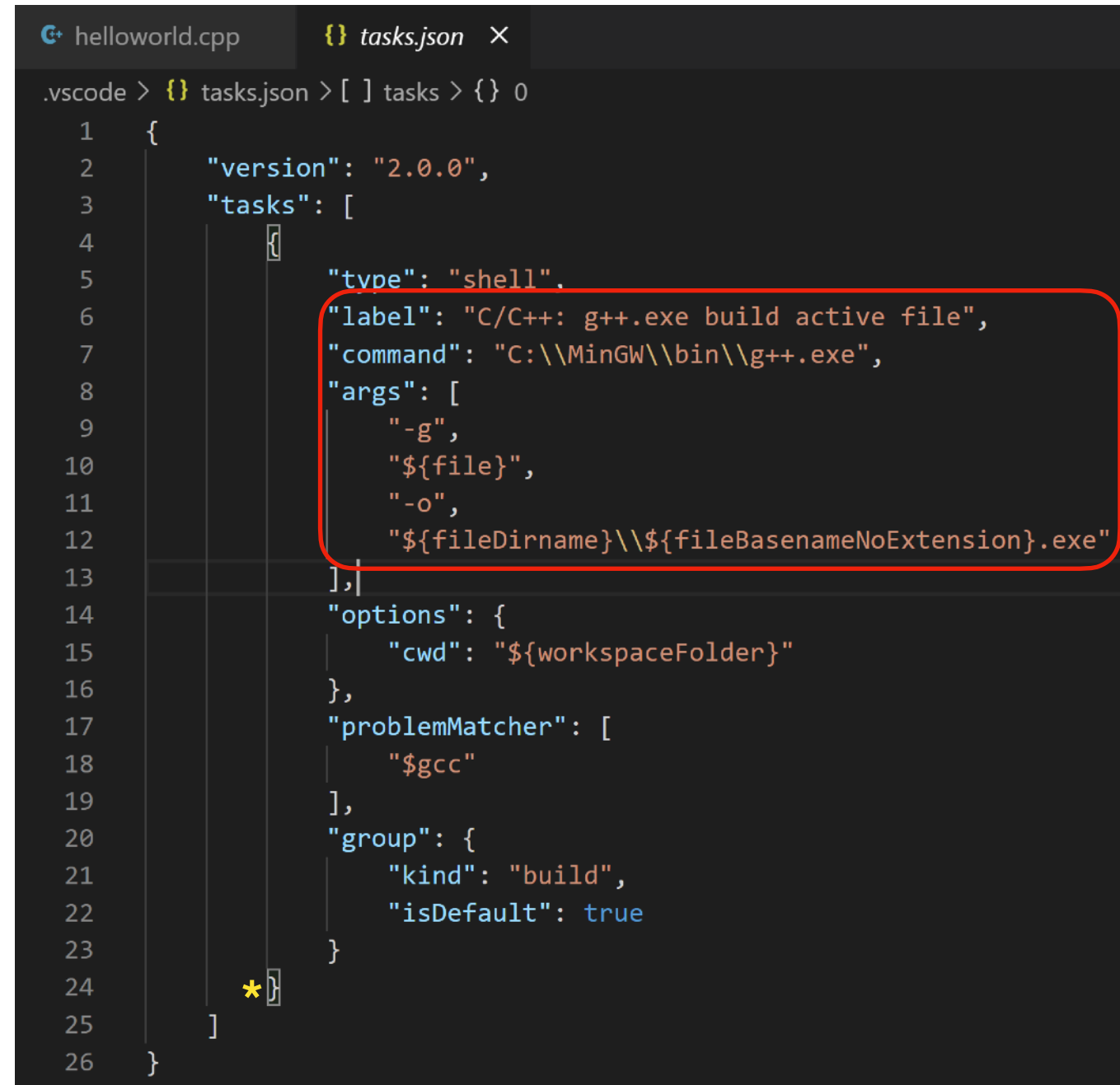
빌드 및 실행 설정(2)



- 나타난 리스트는 환경에 따라 조금씩 다를 수 있습니다.
- g++.exe가 포함되어 있는 설정을 선택.
- 예시의 두번째는 선택된(active) 파일을 g++.exe를 이용하여 빌드 하는 경우의 task를 정리하는 파일입니다.

빌드 및 실행 설정(3)

- 자동으로 .vscode 폴더에 tasks.json이 생성되어 열립니다.
- PATH 설정이 잘 되어있다면 오른쪽과 같은 내용이 들어 있습니다.
- label은 이전의 리스트에 표시되는 이름이고, command는 실제 컴파일러의 파일경로입니다.
- args는 컴파일러 실행시의 argument들을 나타냅니다.
 - -g: 디버깅 정보 포함
 - \${file}: 선택된 파일이름
 - -o \${fileDirname}~~~: 컴파일 후 생성될 실행파일의 이름

A screenshot of the Visual Studio Code editor showing the tasks.json file. The file is located in the .vscode folder. The JSON content is as follows:

```
.vscode > {} tasks.json > [ ] tasks > {} 0
1  {
2      "version": "2.0.0",
3      "tasks": [
4          {
5              "type": "shell",
6              "label": "C/C++: g++.exe build active file",
7              "command": "C:\\MinGW\\bin\\g++.exe",
8              "args": [
9                  "-g",
10                 "${file}",
11                 "-o",
12                 "${fileDirname}\\${fileBasenameNoExtension}.exe"
13             ],
14              "options": {
15                  "cwd": "${workspaceFolder}"
16              },
17              "problemMatcher": [
18                  "$gcc"
19              ],
20              "group": {
21                  "kind": "build",
22                  "isDefault": true
23              }
24          }
25      ]
26  }
```

The task configuration on lines 4-13 is highlighted with a red box. The file has a yellow asterisk icon at the bottom, indicating it is unsaved.

빌드 및 실행 설정(4)

- 자동 생성된 설정은 빌드에 대한 것 뿐입니다.
- 편의를 위해 tasks 아래에 (앞 슬라이드의 '*'표시 확인) 프로그램 실행을 위한 task를 하나 더 추가합니다.
- Mac/Linux를 사용하는 학생들은 command를 아래의 내용으로 바꾸고 args를 빼면됩니다.
- 강의 GitHub의 practice1에 있는 tasks.json을 복사해와도 됩니다.

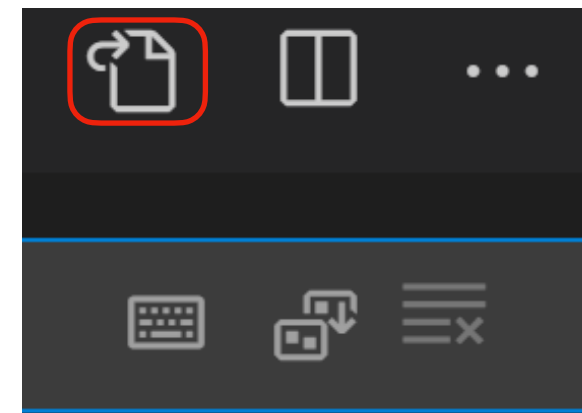
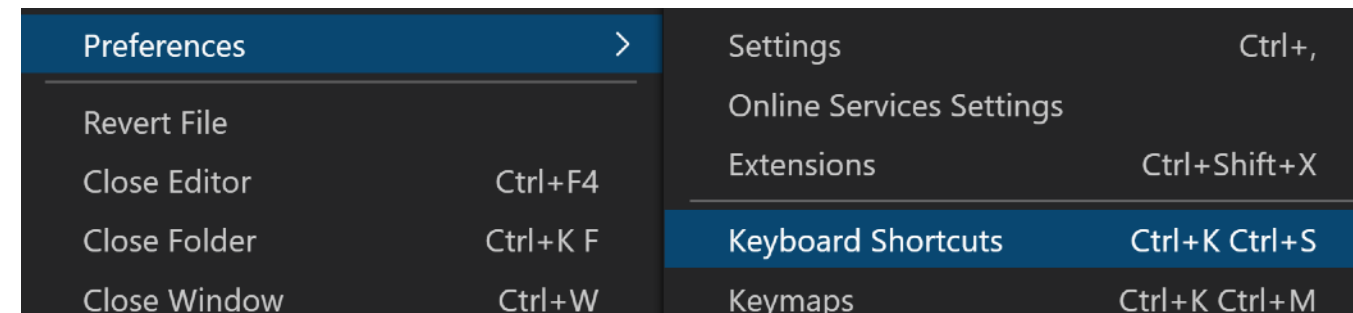
```
    "kind": "build",  
    "isDefault": true  
  },  
  *},  
  {  
    "label": "Run C++",  
    "command": "cmd",  
    "args": [  
      "/C",  
      "${fileDirname}\\${fileBasenameNoExtension}.exe"  
    ],  
    "group": {  
      "kind": "test",  
      "isDefault": true  
    }  
  }  
]
```

For Mac/Linux

```
"command": "${fileDirname}/${fileBasenameNoExtension}.exe",
```

빌드 및 실행 설정(5)

- 다음은 단축키 설정입니다.
- Ctrl+K, Ctrl+S를 연속으로 입력하면 **Keyboard Shortcut** 설정화면이 열립니다.
- 또는 오른쪽처럼 메뉴를 선택해도 됩니다.
- 나타난 화면에서 빨간색으로 표시된 것과 같은 아이콘을 찾아 클릭합니다.



빌드 및 실행 설정(6)

```
// Place your key bindings in this file to override the defaults
[
  //Build
  { "key": "ctrl+alt+b", "command": "workbench.action.tasks.build" },
  //Run
  { "key": "ctrl+alt+r", "command": "workbench.action.tasks.test" }
]
```

- 다시 단축키 설정을 저장하는 JSON 파일이 열립니다.
- 위에 나온 내용을 [] 사이에 입력하고 저장하세요.
- 첫번째는 빌드를 위한 단축키, 두번째는 실행을 위한 단축키입니다.
- 강의 GitHub에서 keybindings.json의 내용을 복사해와도 됩니다.

Hello World

- 드디어 긴 설정이 끝나고 첫 프로그램을 작성할 시간입니다.
- 오른쪽과 같은 내용을 helloworld.cpp 파일에 입력하세요.
- 강의 GitHub에서 내용을 복사해도 됩니다.

```
helloworld.cpp X
practice1 > helloworld.cpp > main()
1  #include<iostream>
2  using namespace std;
3
4  int main() {
5      cout << "Hello World!!\n";
6      return 0;
7  }
```

Hello World 빌드

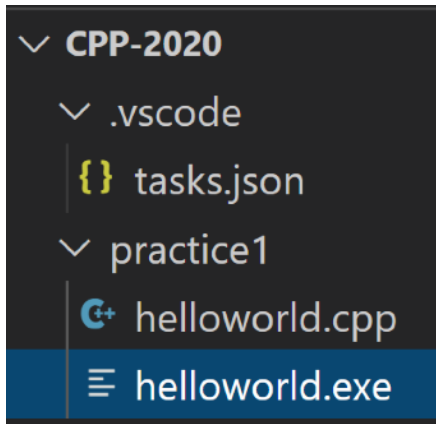
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
> Executing task: C:\MinGW\bin\g++.exe -g c:\cpp-2020\practice1\helloworld.cpp -  
o c:\cpp-2020\practice1\helloworld.exe <
```

Terminal will be reused by tasks, press any key to close it.

- 이제 빌드를 하여 실행파일을 만들어 봅시다.
- helloworld.cpp를 선택합니다.
- 이전에 설정한 단축키 ctrl+alt+B를 눌러 빌드를 실행합니다.
- 위와 같이 g++ 컴파일러가 실행되는 화면이 밑의 TERMINAL에 표시되면 성공입니다.

Hello World 실행



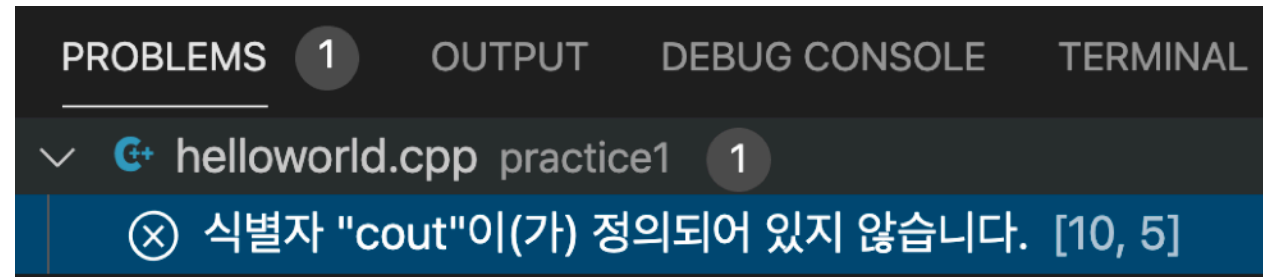
```
> Executing task: C:\Windows\system32\cmd.exe /C c:\cpp-2020\practice1\helloworld.exe <

Hello World!!

Terminal will be reused by tasks, press any key to close it.
```

- 빌드가 성공적이었다면 helloworld.exe 파일이 생성됩니다.
- helloworld.exe를 선택하고 설정한 실행 단축키인 ctrl+alt+r을 누르면 프로그램이 실행됩니다.
- 위의 오른쪽과 같이 “Hello World!!”가 출력되면 성공입니다.
- Hello World!가 출력된 화면을 캡처하여 출석확인용으로 제출하세요.

에러 확인방법



- 코드 작성 중, 뭔가 잘못된 부분이 있다면 VSCode 하단의 PROBLEMS탭에 문제와 위치가 표시됩니다.
- 또는 빌드를 실행하고 나면 TERMINAL 탭에 컴파일 에러 메시지가 표시됩니다.

```
/Users/jindae/vscode-workspace/cpp-2020/practice1/helloworld.cpp:10:5:
error:
    use of undeclared identifier 'cout'; did you mean 'std::cout'?
    cout << "Hello World!!\n";
    ^~~~~
    std::cout
/Library/Developer/CommandLineTools/usr/include/c++/v1/iostream:54:33:
note:
    'std::cout' declared here
extern _LIBCPP_FUNC_VIS ostream cout;
1 error generated.
```

실습 정리

- 이로써 VSCode에서 Hello World 프로그램을 실행하는 실습을 모두 마쳤습니다.
- 오늘 설정한 개발환경은 앞으로의 실습에도 계속해서 사용됩니다.
- 차후 실습에서는 만들어진 workspace에서 추가로 폴더와 cpp파일을 만들고 빌드/실행을 보다 간단하게 할 수 있습니다.
- 개발환경 구축방법은 C++ 프로그램 개발의 기반이 되는 환경을 구축하는 방법이므로 잘 기억해두는 것이 좋습니다.
- 온라인에서 보다 편리한 개발환경을 위한 추가적인 다양한 설정을 찾아볼 수 있습니다.