

Lab 0

Introduction of Programming Environment

DCSLAB 안병철 / 2023-03-08

Exams, Exercises and Term Projects

- **Midterm** (25%) = 4/25 (Tue) or 4/27 (Thu) | **Final** (25%)= 6/13 (Tue)
- **Exercises** (10%) + **Two Term Projects** (30%)
- Attendance (10%)

- Each exercise is **due by 18:00** on the day of the next lab session.
 - E.g. Exercises in Lab0(3/8) are due by 18:00 on the day of Lab1(3/15)




- Exercise will be helpful for your term projects.
- Term Projects = C++ related & Java related

Exercise 1

- 3~4 Simple C programming review problems + 1 slightly tricky problem
 - Print String / Integer and Floating point Arithmetic / Functions / Pointer + struct mixed with pointer
- Skeletons will be provided
- You must submit a solution for every problem
- Fill the skeletons with **texts specified in the guide** if you can't solve the problems
- **!! Follow the guide !!**

C++ Programming Environment Setup

- Visual Studio Code for C++ Programming
- <https://code.visualstudio.com/download>



↓ Windows
Windows 8, 10, 11

User Installer [x64](#) [x86](#) [Arm64](#)
System Installer [x64](#) [x86](#) [Arm64](#)
.zip [x64](#) [x86](#) [Arm64](#)
CLI [x64](#) [x86](#) [Arm64](#)

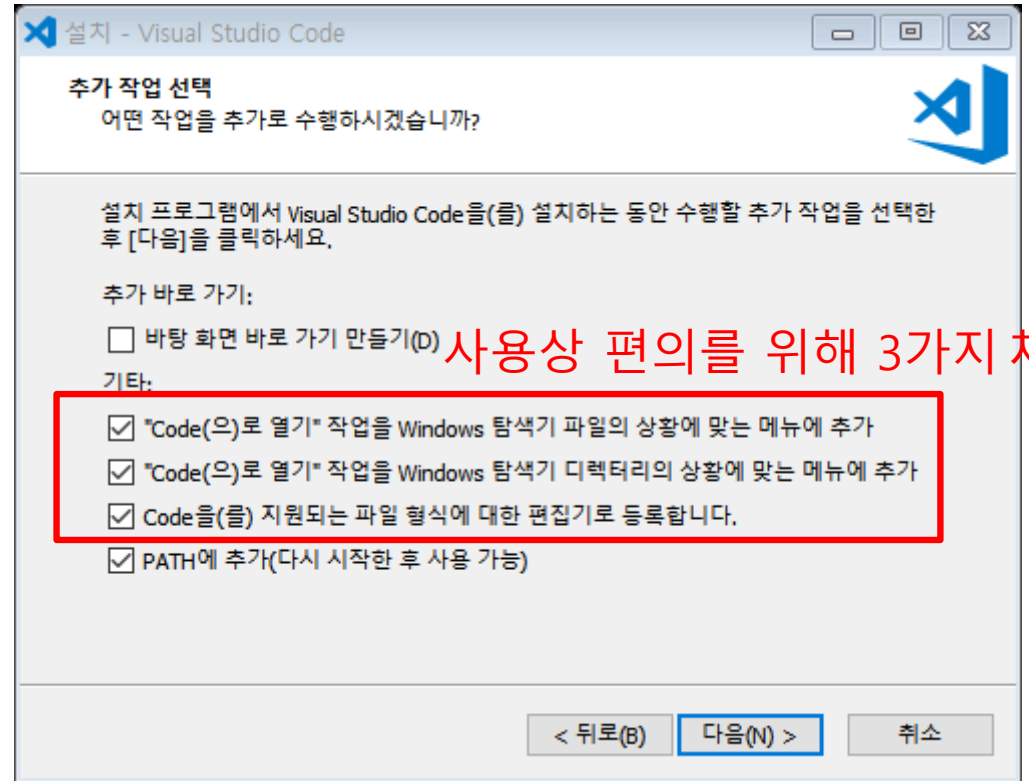
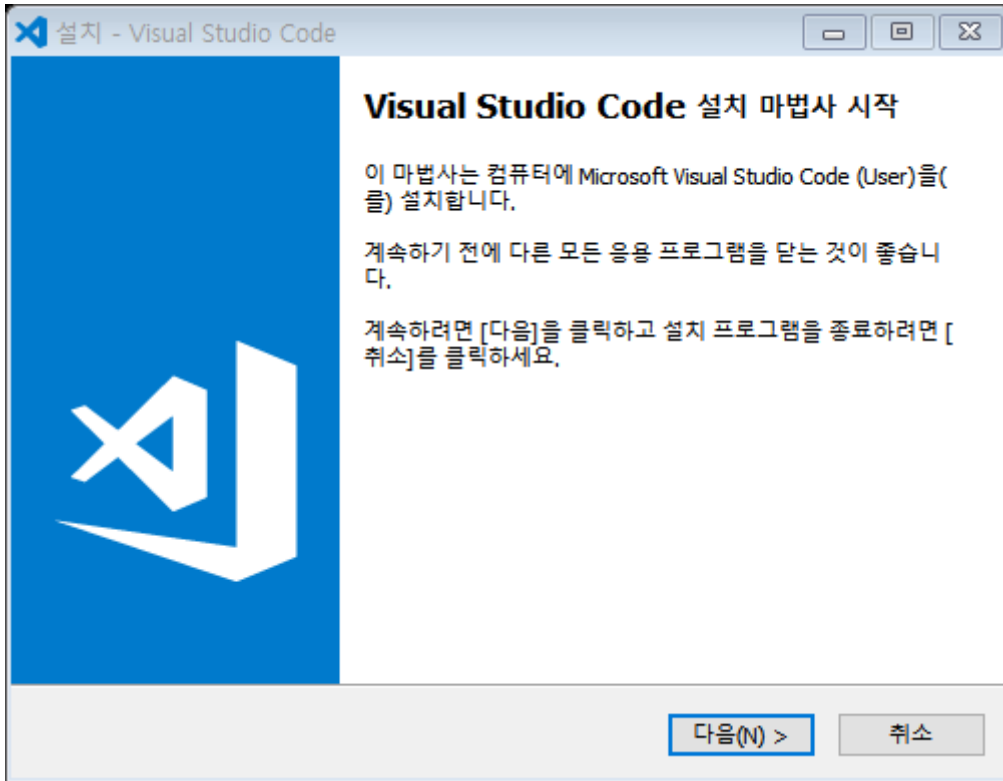
↓ .deb ↓ .rpm
Debian, Ubuntu Red Hat, Fedora, SUSE

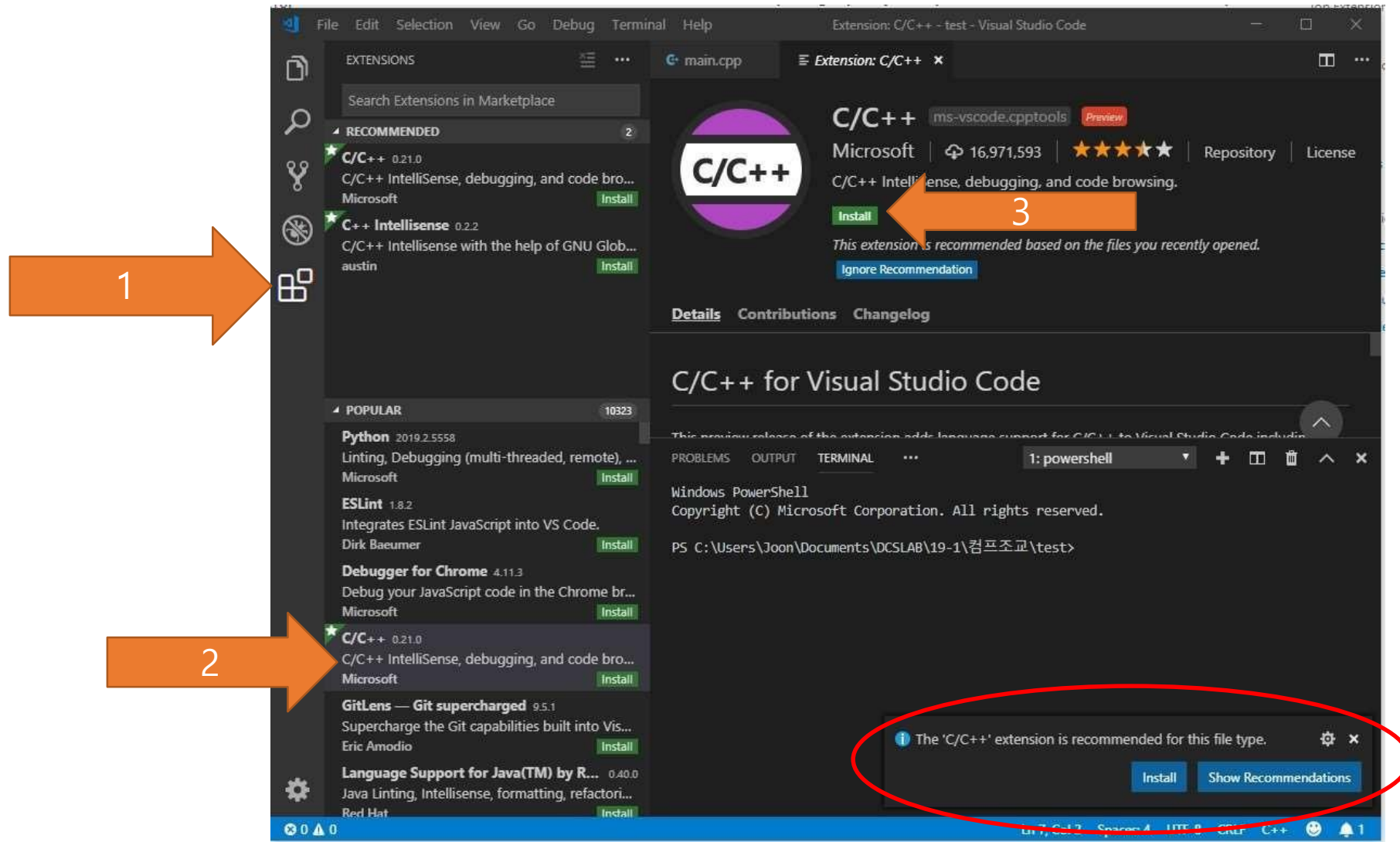
.deb [x64](#) [Arm32](#) [Arm64](#)
.rpm [x64](#) [Arm32](#) [Arm64](#)
.tar.gz [x64](#) [Arm32](#) [Arm64](#)
Snap [Snap Store](#)
CLI [x64](#) [Arm32](#) [Arm64](#)

↓ Mac
macOS 10.11+

.zip [Intel chip](#) [Apple silicon](#) [Universal](#)
CLI [Intel chip](#) [Apple silicon](#)

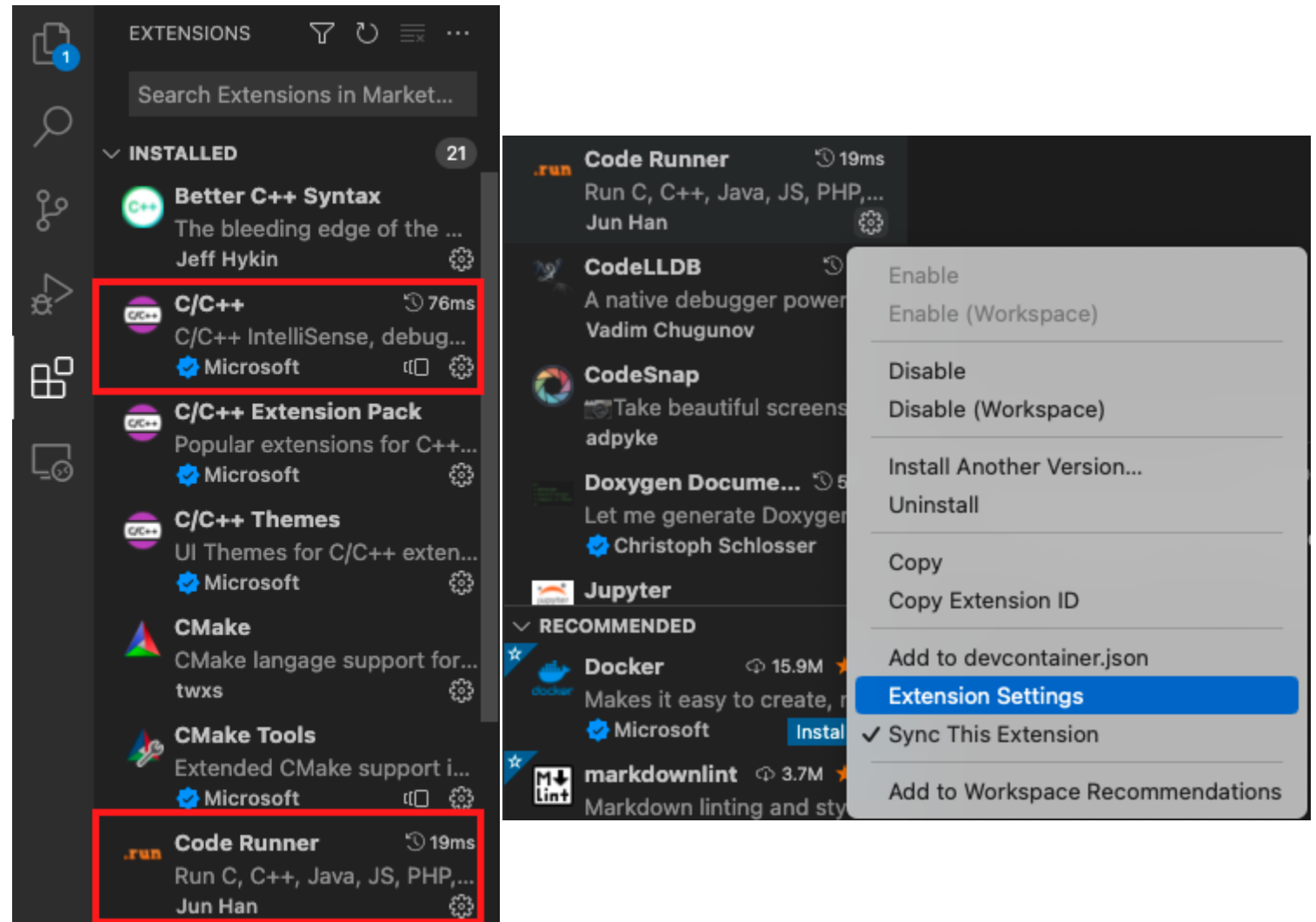
C++ Programming Environment Setup (For Windows)





For MacOS

- For MacOS
- Install "Code Runner"
- Enter Code Runner's "Extension Settings"



For MacOS

- Check "Run In Terminal"
- Click "Edit in settings.json" under "Executor Map"

The screenshot shows the VS Code settings interface for the Code Runner extension. The search bar at the top contains the text "@ext:formulahendry.code-runner" and indicates "23 Settings Found". The settings are organized into sections: "Code-runner: Preserve Focus" (checked), "Code-runner: Respect Shebang" (checked), "Code-runner: Run In Terminal" (unchecked), and "Code-runner: Save All Files Before Run" (unchecked). The bottom of the window shows the "TERMINAL" tab selected.

This close-up shows the "Code-runner: Executor Map" setting. The text reads "Set the executor of each language." Below this, there is a blue link that says "Edit in settings.json".

For MacOS

- The command might already exist in the JSON file depending on system settings.

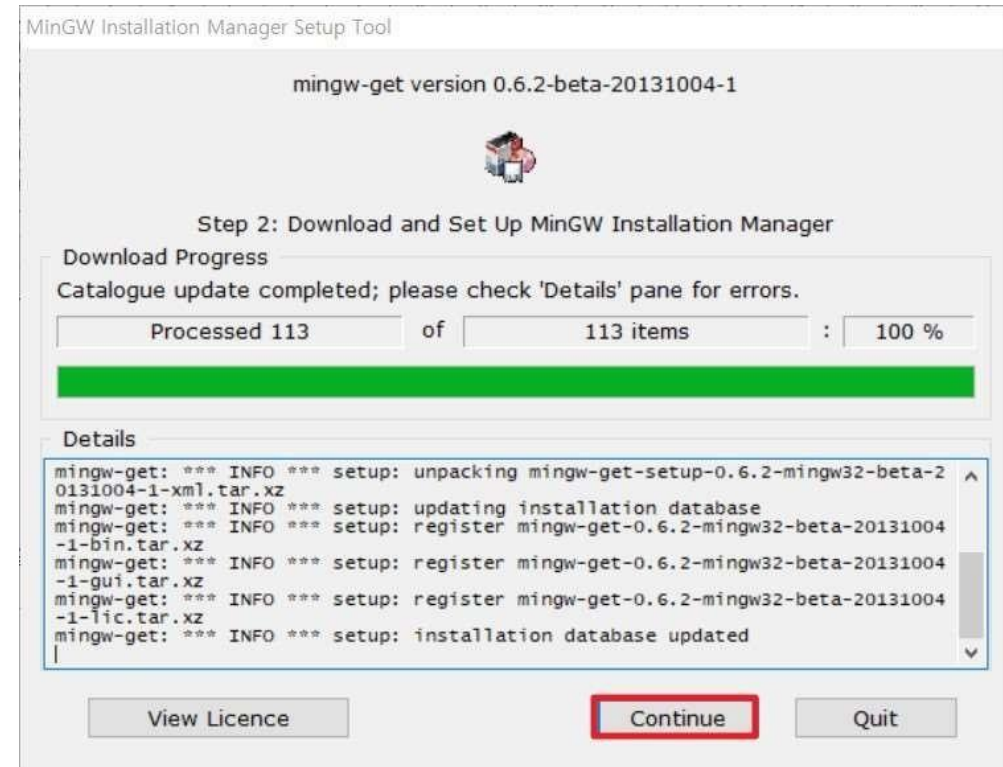
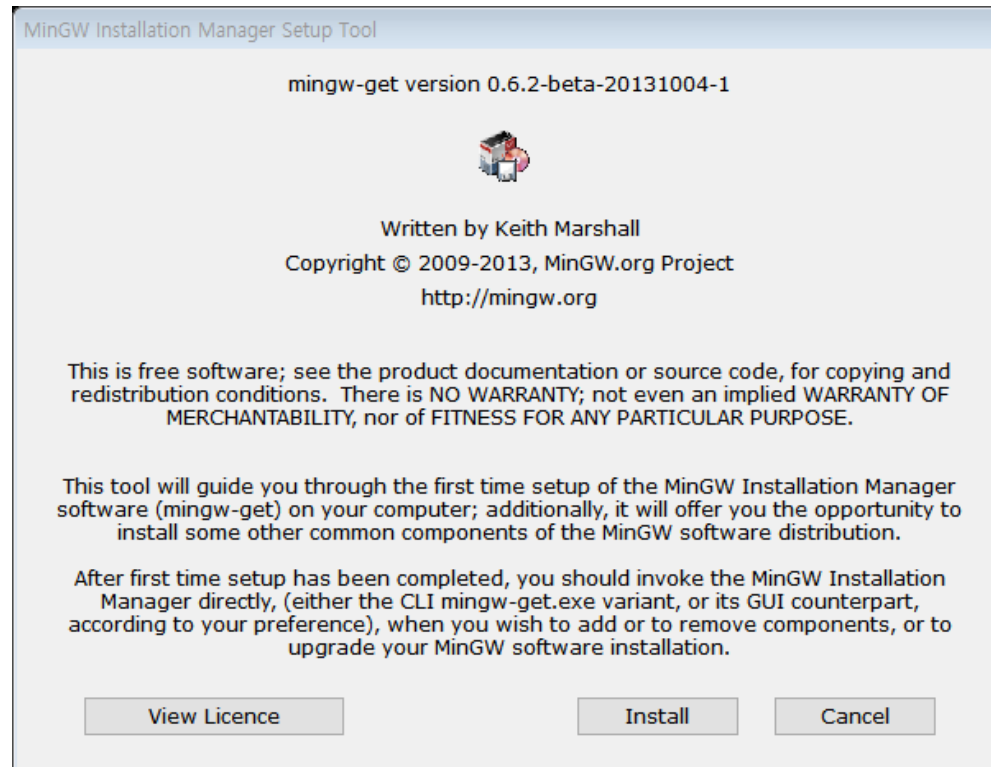
```
3  "code-runner.executorMap": {  
4  
5      "javascript": "node",  
6      "java": "cd $dir && javac $fileName && java $fileNameWithoutExt",  
7      "c": "cd $dir && gcc $fileName -o $fileNameWithoutExt && $dir$fi  
8      "zig": "zig run",  
9      "cpp": "cd $dir && g++ $fileName -o $fileNameWithoutExt && $dir$-  
10     "objective-c": "cd $dir && gcc -framework Cocoa $fileName -o $fi  
11     "php": "php",  
12     "python": "python -u",  
13     "perl": "perl",  
14     "perl6": "perl6",  
15     "ruby": "ruby",
```

- Add the following command to the json file

```
"code-runner.executorMap": {  
    "c": "cd $dir && gcc $fileName -o $fileNameWithoutExt && $dir$fileNameWithoutExt",  
    "cpp": "cd $dir && g++ -std=c++20 $fileName -o $fileNameWithoutExt && $dir$fileNameWithoutExt"  
}
```

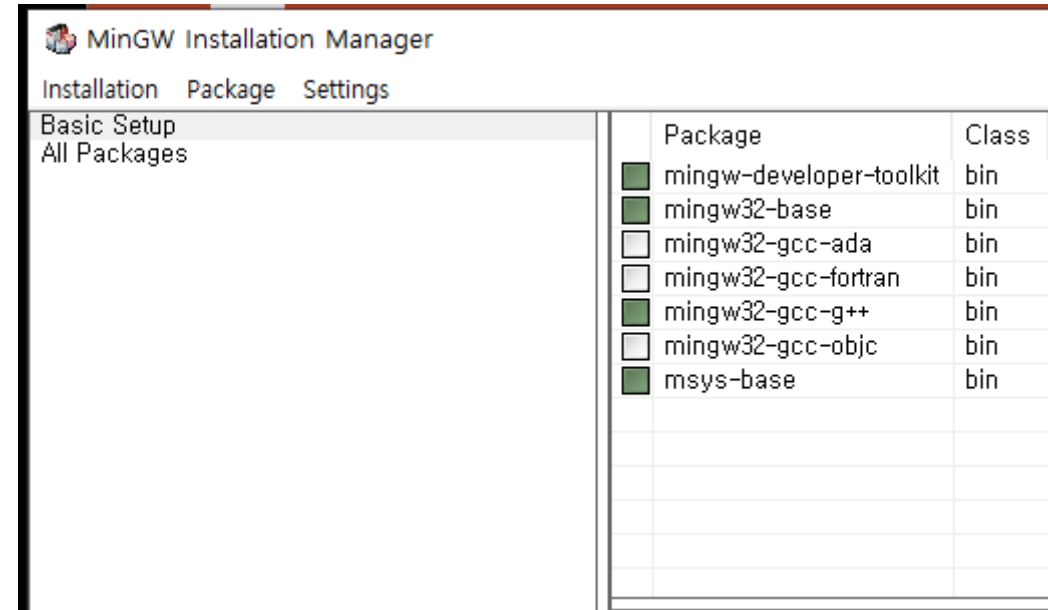
C++ Programming Environment Setup (For Windows)

- For gcc compile, setup mingw
- <https://sourceforge.net/projects/mingw/>







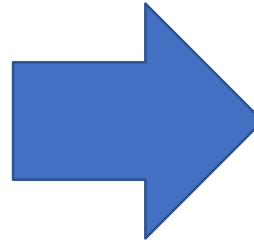
C++ Programming Environment Setup (For Windows)

- Check package
mingw-developer-toolkit,
mingw32-base, mi
ngw32-gcc-g++,
msys-base-bin

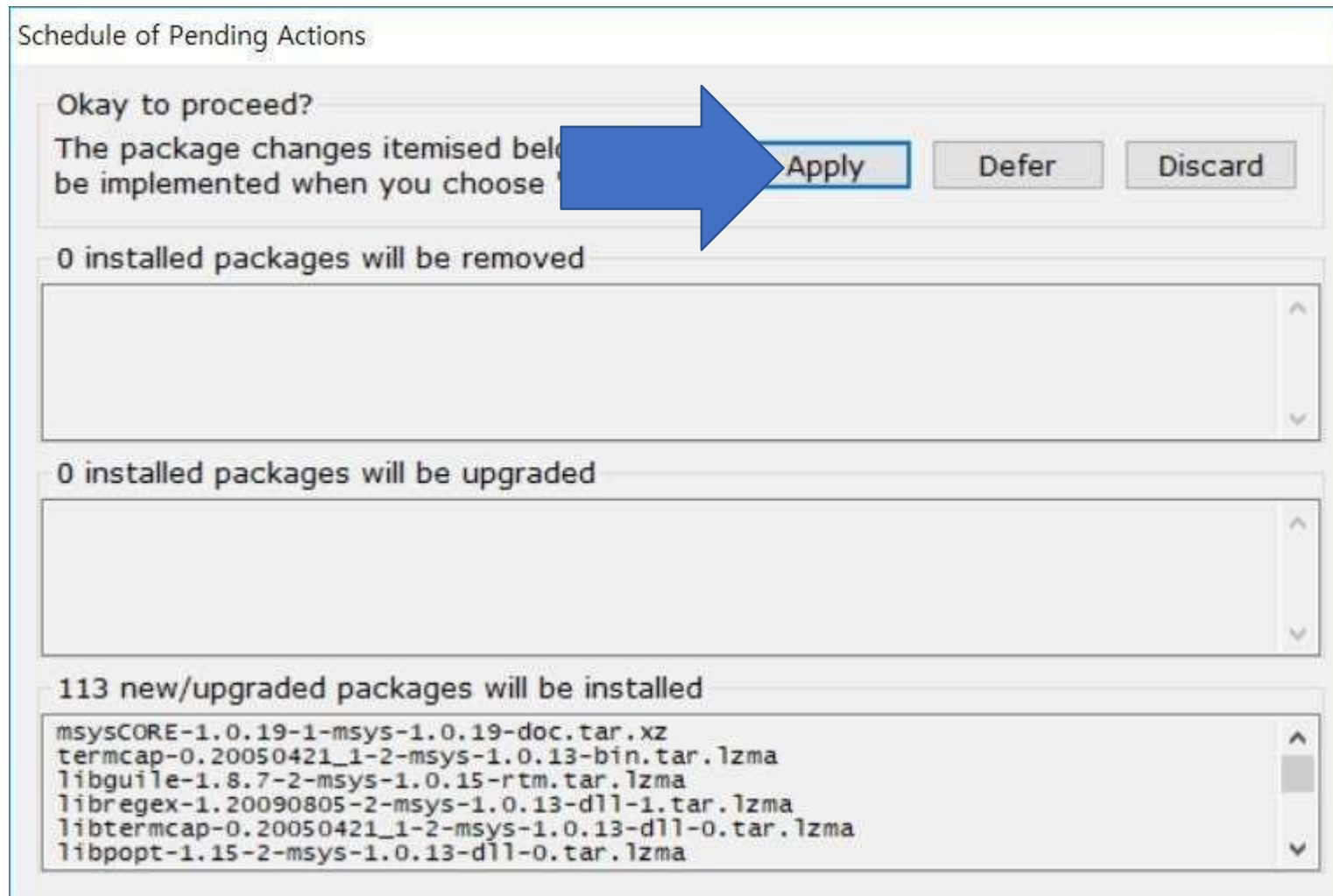


C++ Programming Environment Setup (For Windows)

| Package | Class | Installe |
|---|-------|----------|
|  mingw-developer-toolkit | bin | |
|  mingw32-base | bin | |
| <input type="checkbox"/> mingw32-gcc-ada | bin | |
| <input type="checkbox"/> mingw32-gcc-fortran | bin | |
|  mingw32-gcc-g++ | bin | |
| <input type="checkbox"/> mingw32-gcc-objc | bin | |
|  msys-base | bin | |

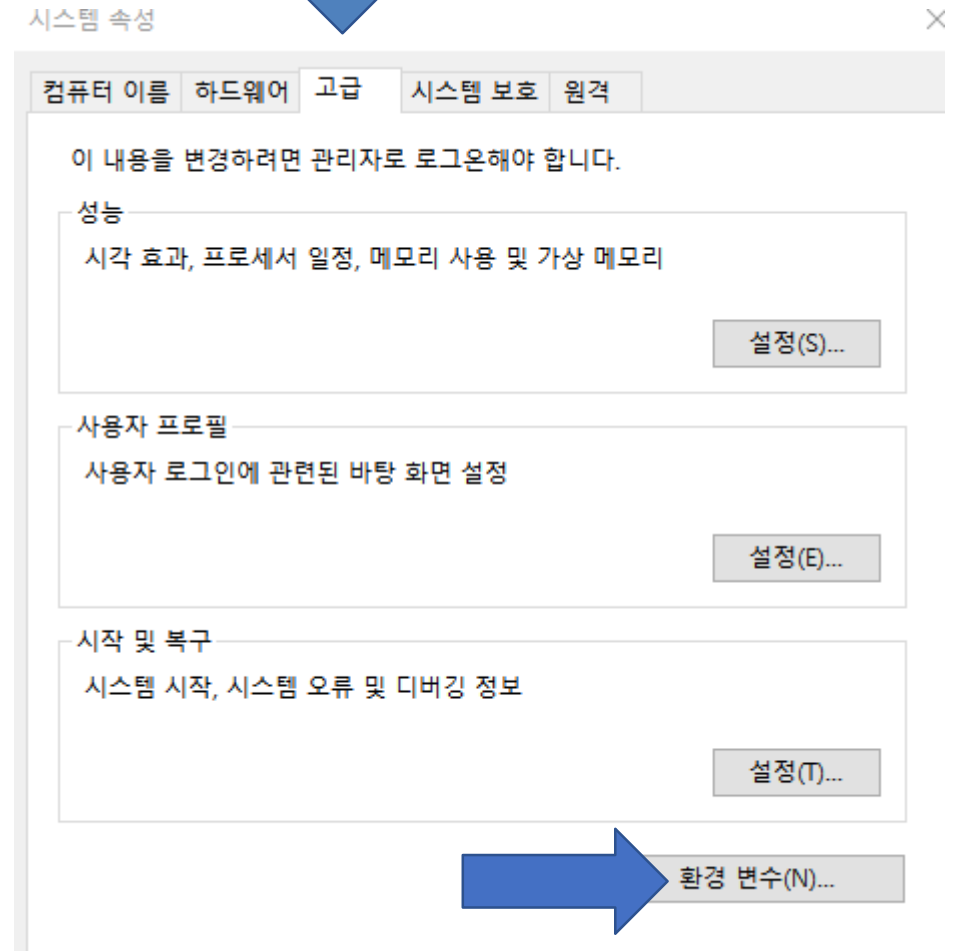
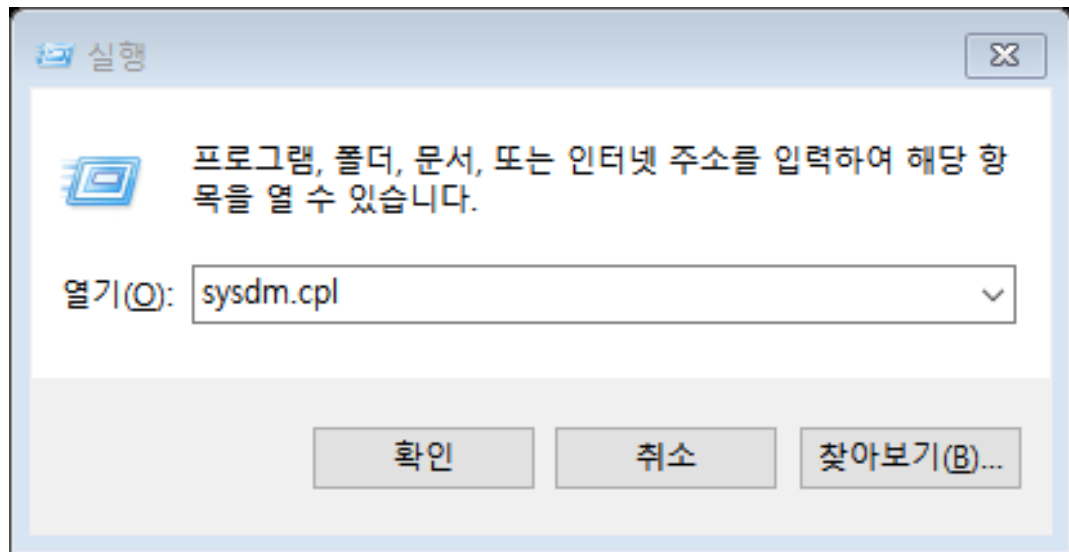


C++ Programming Environment Setup (For Windows)



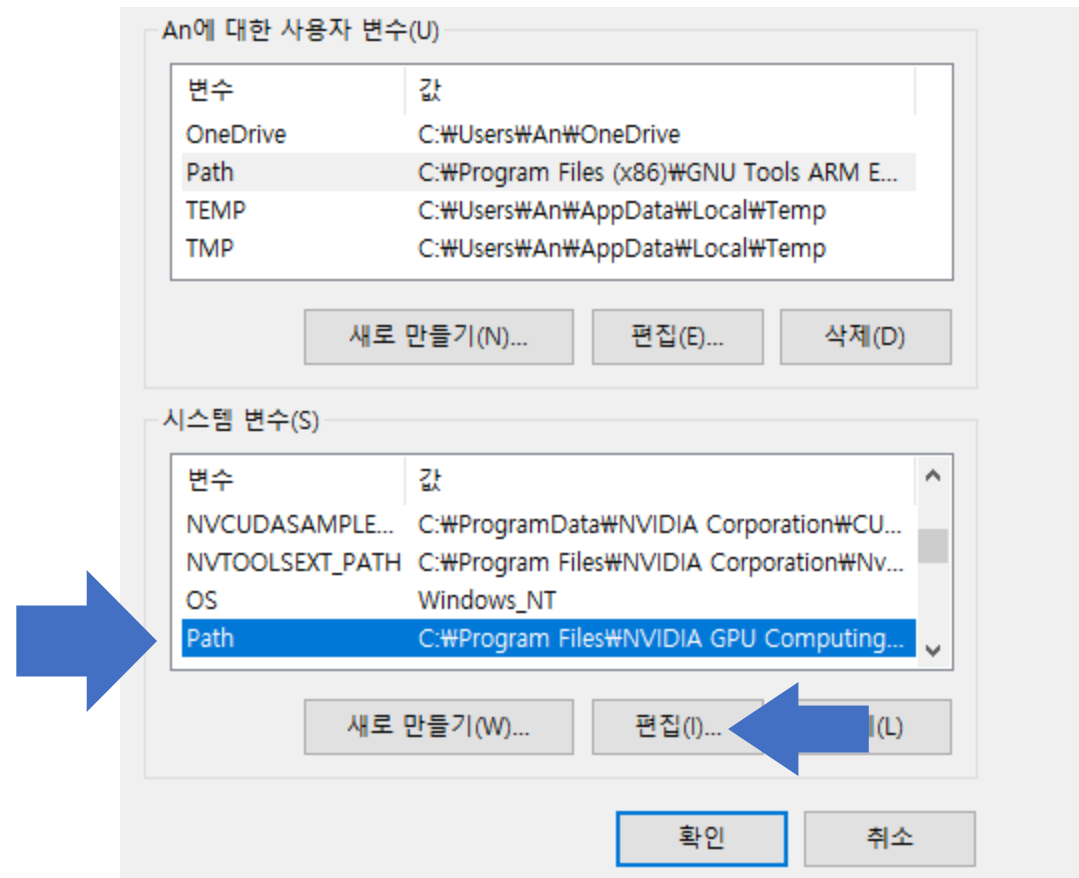
C++ Programming Environment Setup (For Windows)

- Edit system path
- Press the **window key + r**, then type
- **sysdm.cpl** to run system properties in Control Panel.

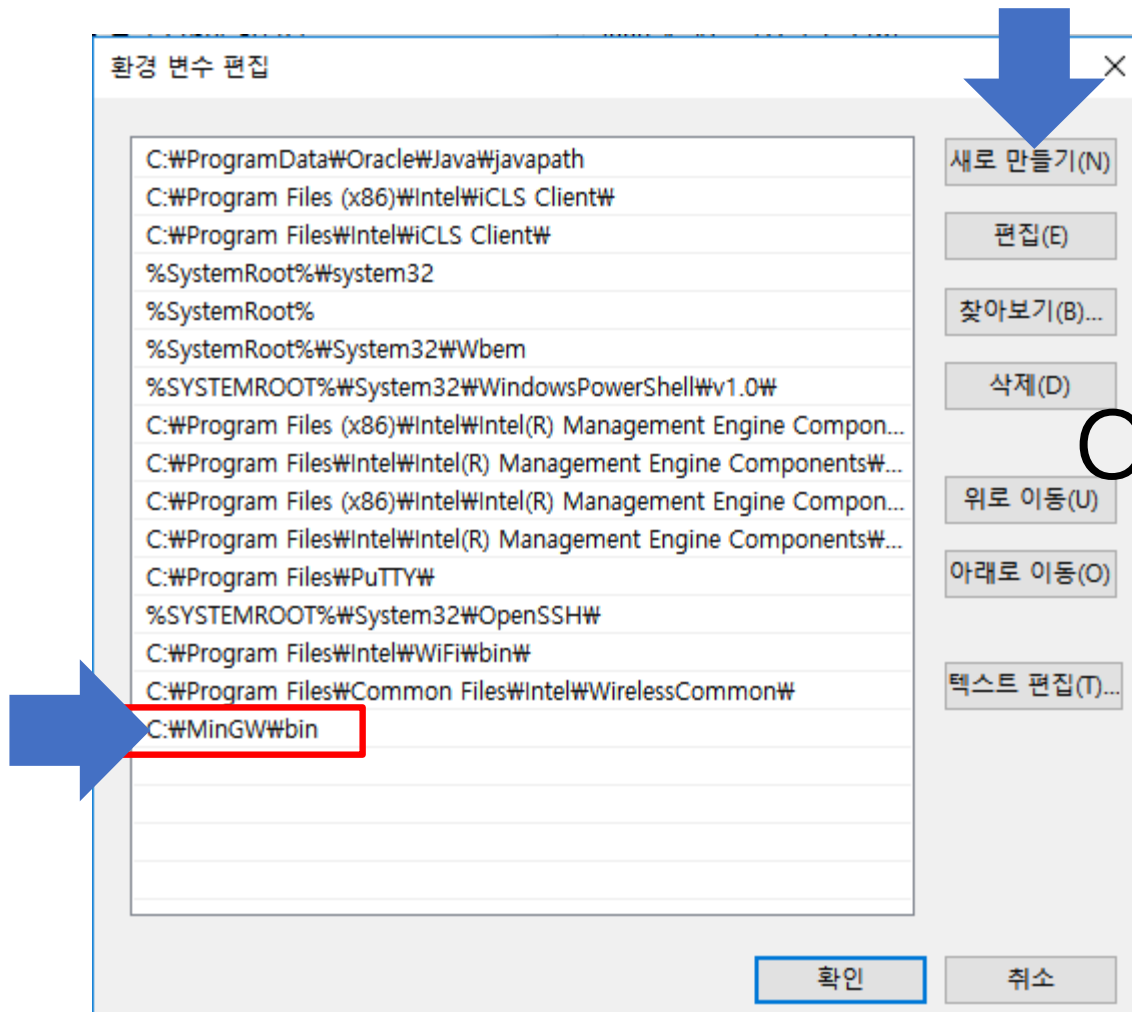


C++ Programming Environment Setup (For Windows)

- Under System Variables, select Path and click the Edit button.

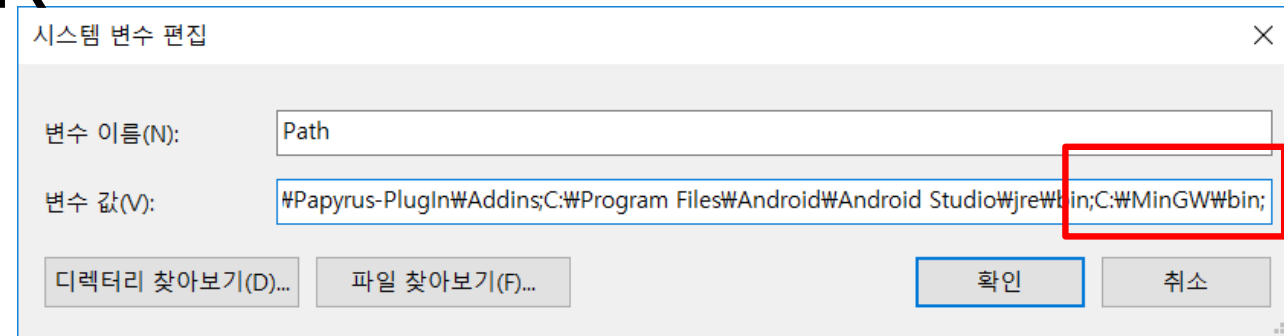


C++ Programming Environment Setup (For Windows)

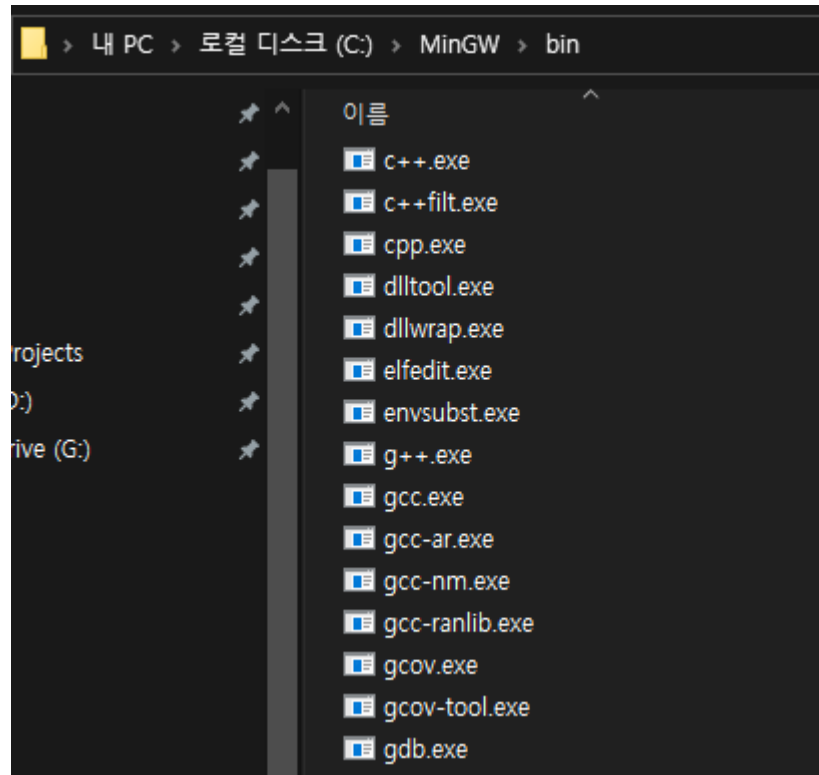


Depending on the version of the window, click New to add the path C:\MinGW\bin or add ;C:\MinGW\bin to the end of the path variable value.

OR

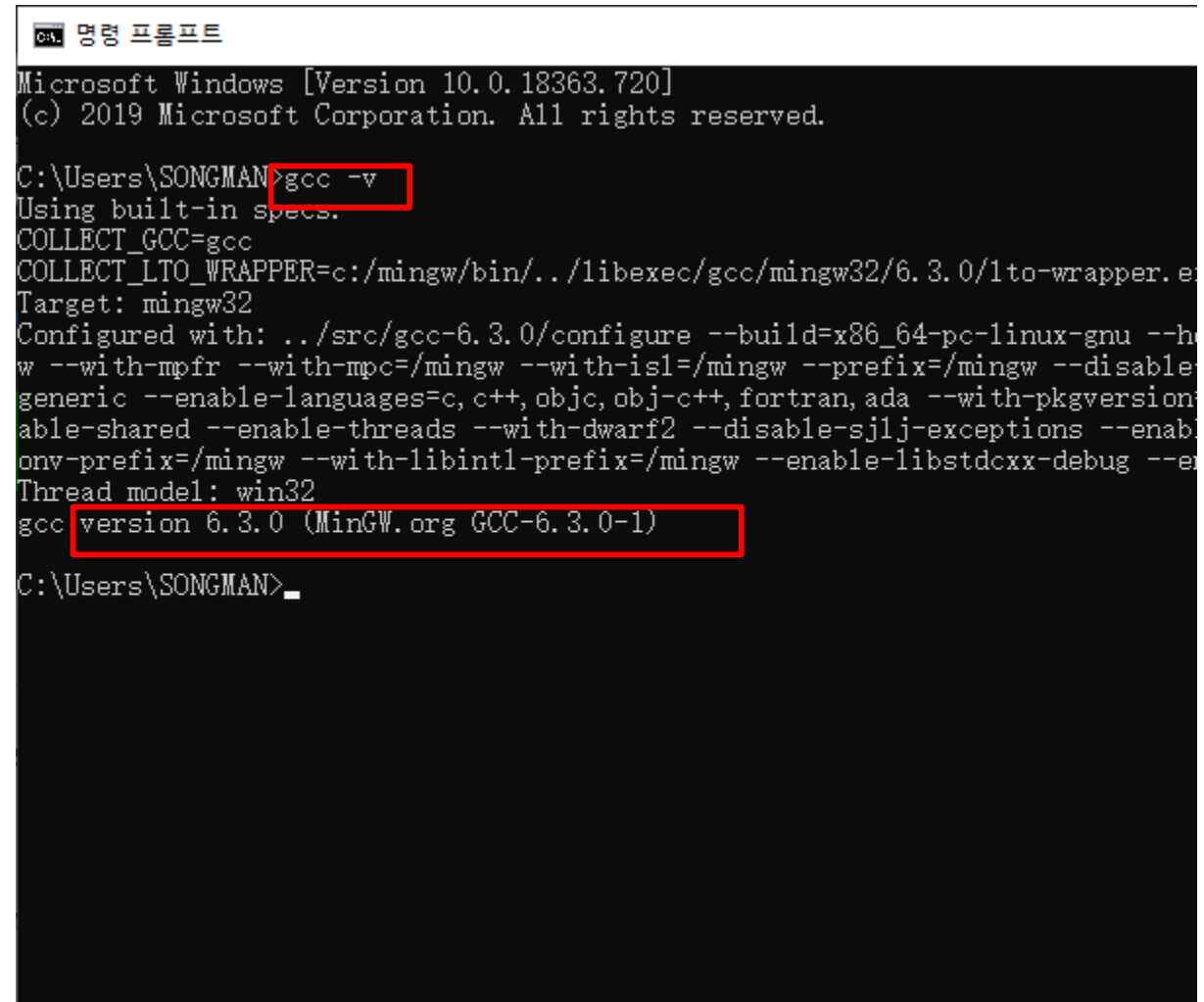
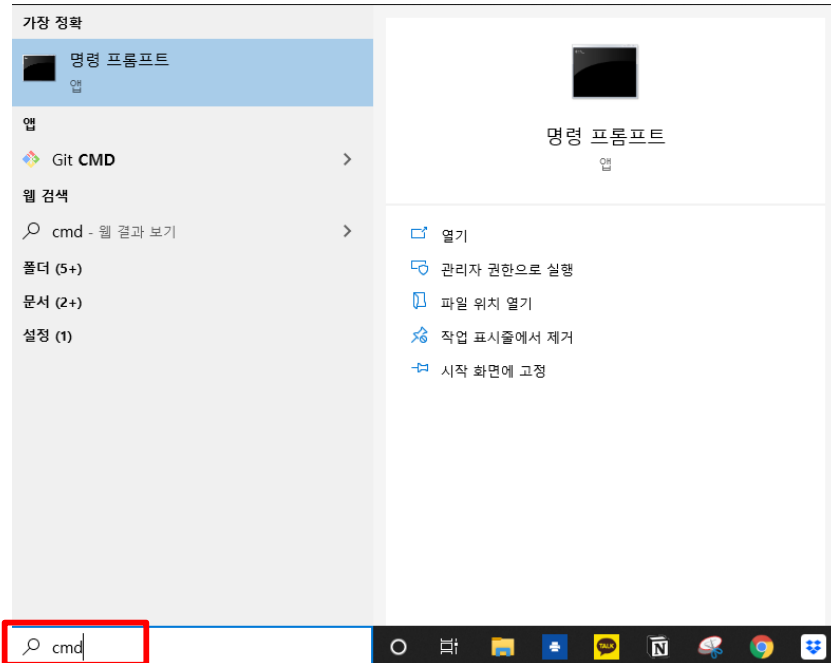


C++ Programming Environment Setup (For Windows)



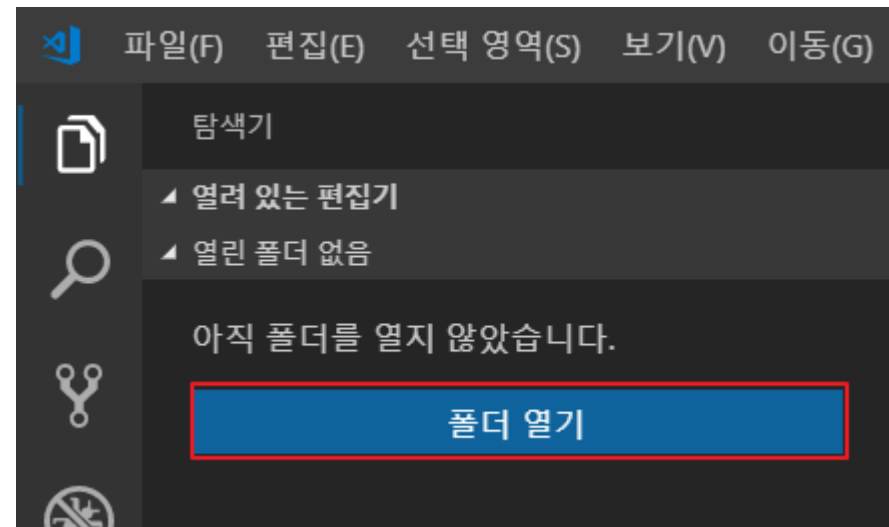
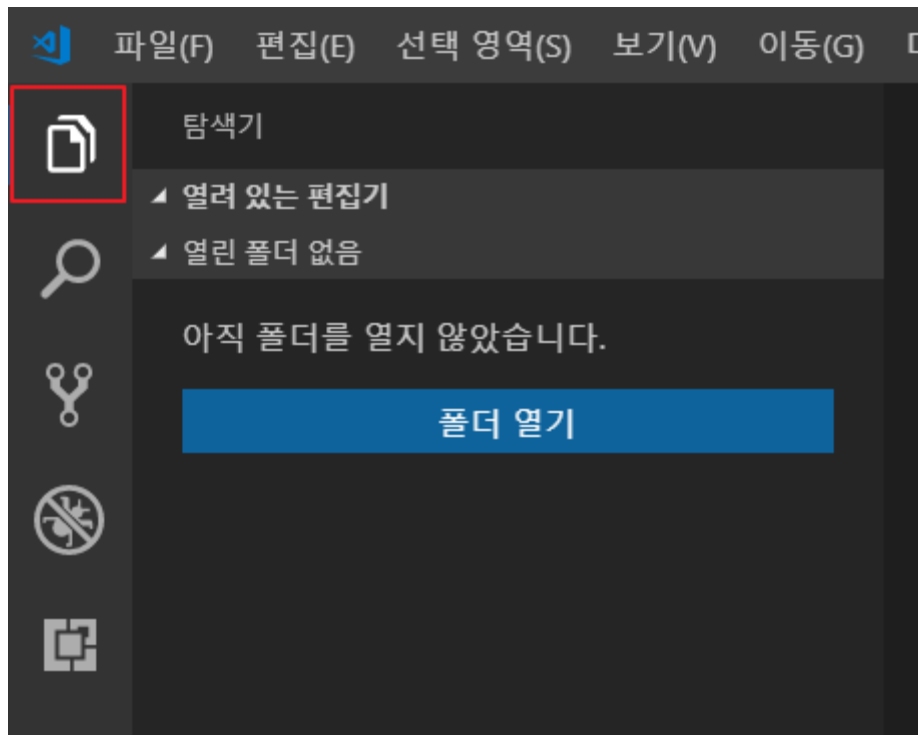
C++ Programming Environment Setup (For Windows)

- If the progress is successful, you can check the gcc, g++ version information at the command prompt as follows:



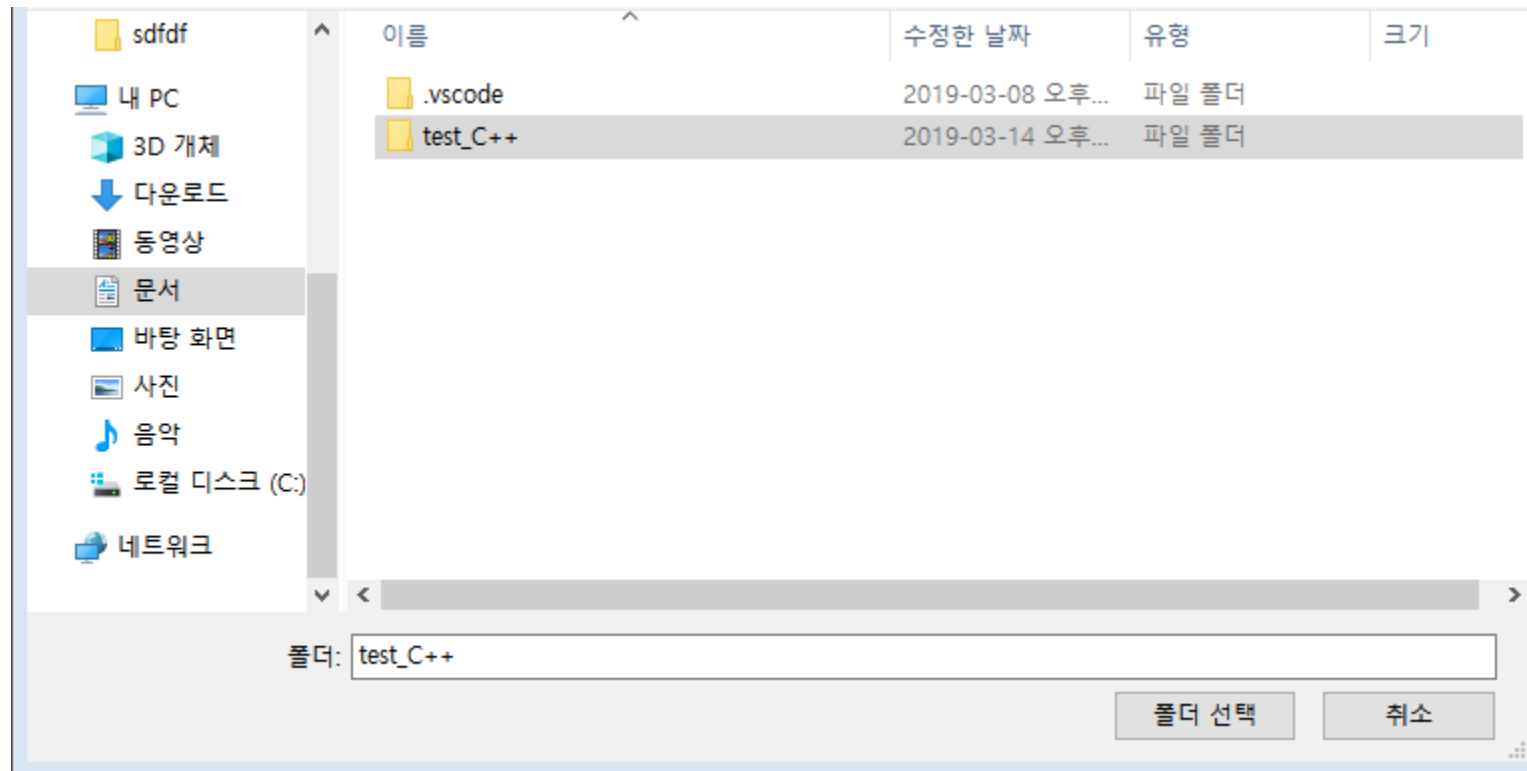
C++ Programming Environment

- Click the Explorer icon in the activity bar located on the left, or press the shortcut Ctrl + Shift + E to open the Explorer on the sidebar as shown in the capture screen below.



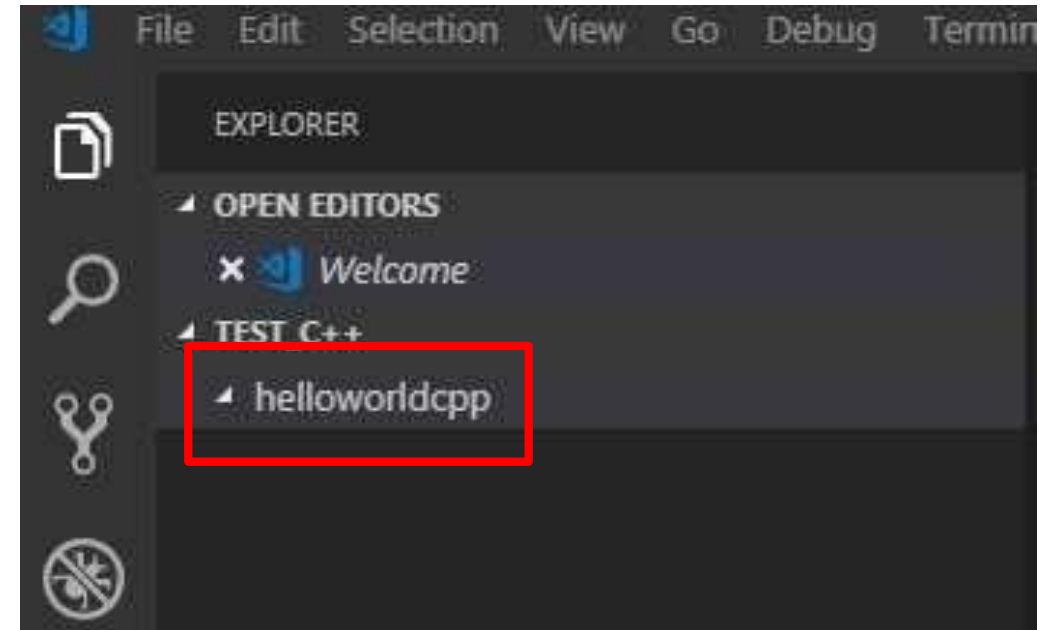
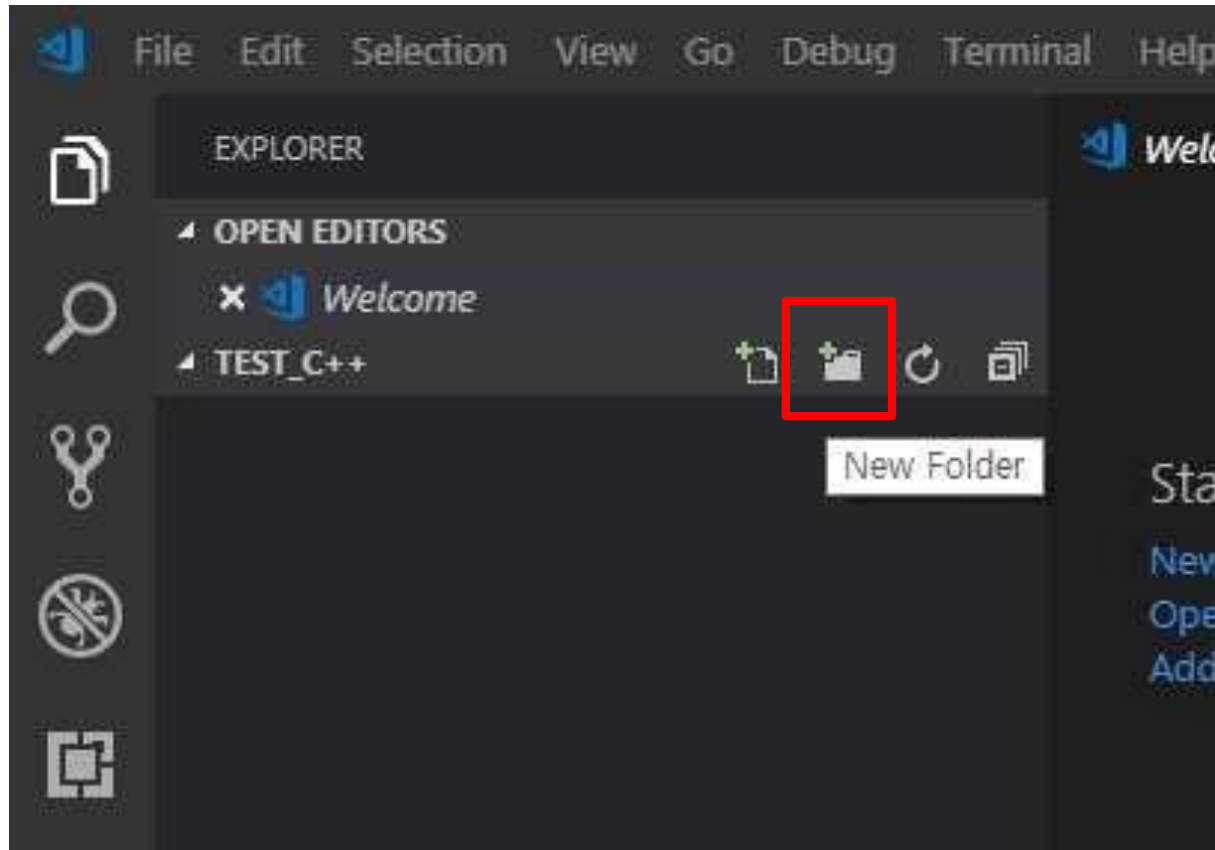
C++ Programming Environment

- Create the test_C++ folder and click the Select Folder button.



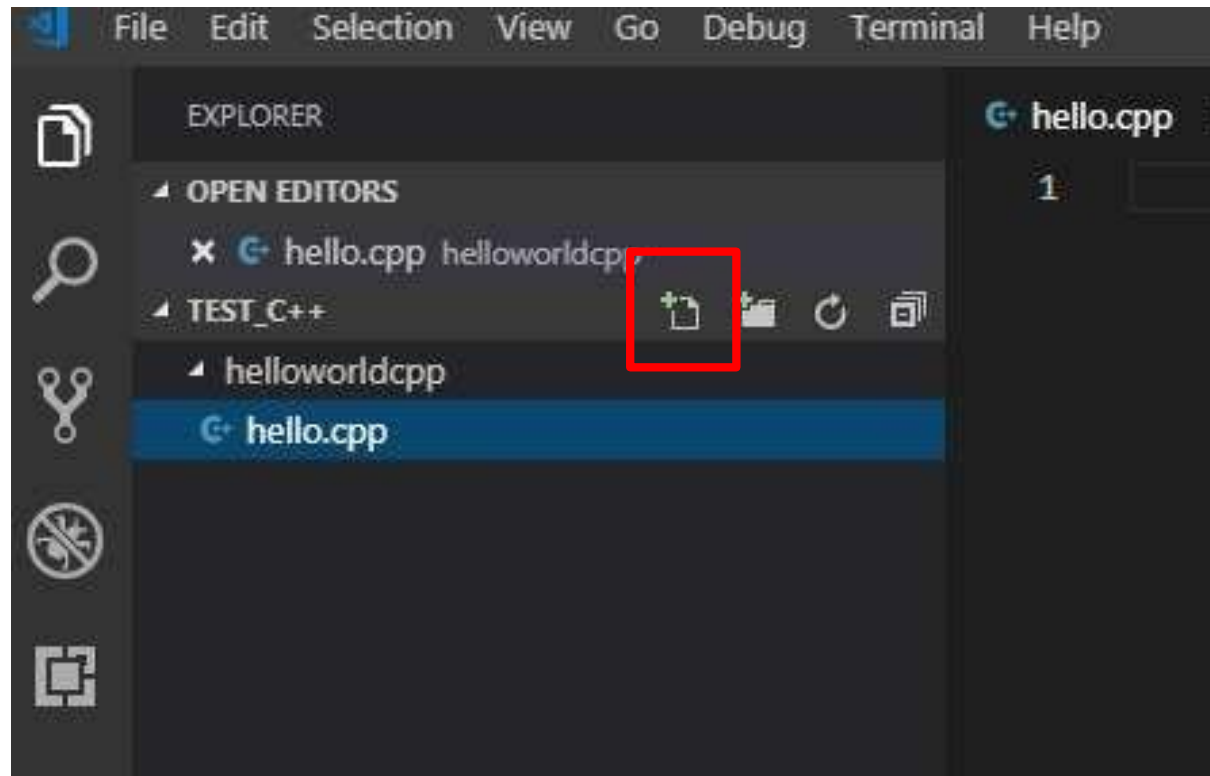
C++ Programming Environment

- Click the new folder icon and create helloworldcpp folder.



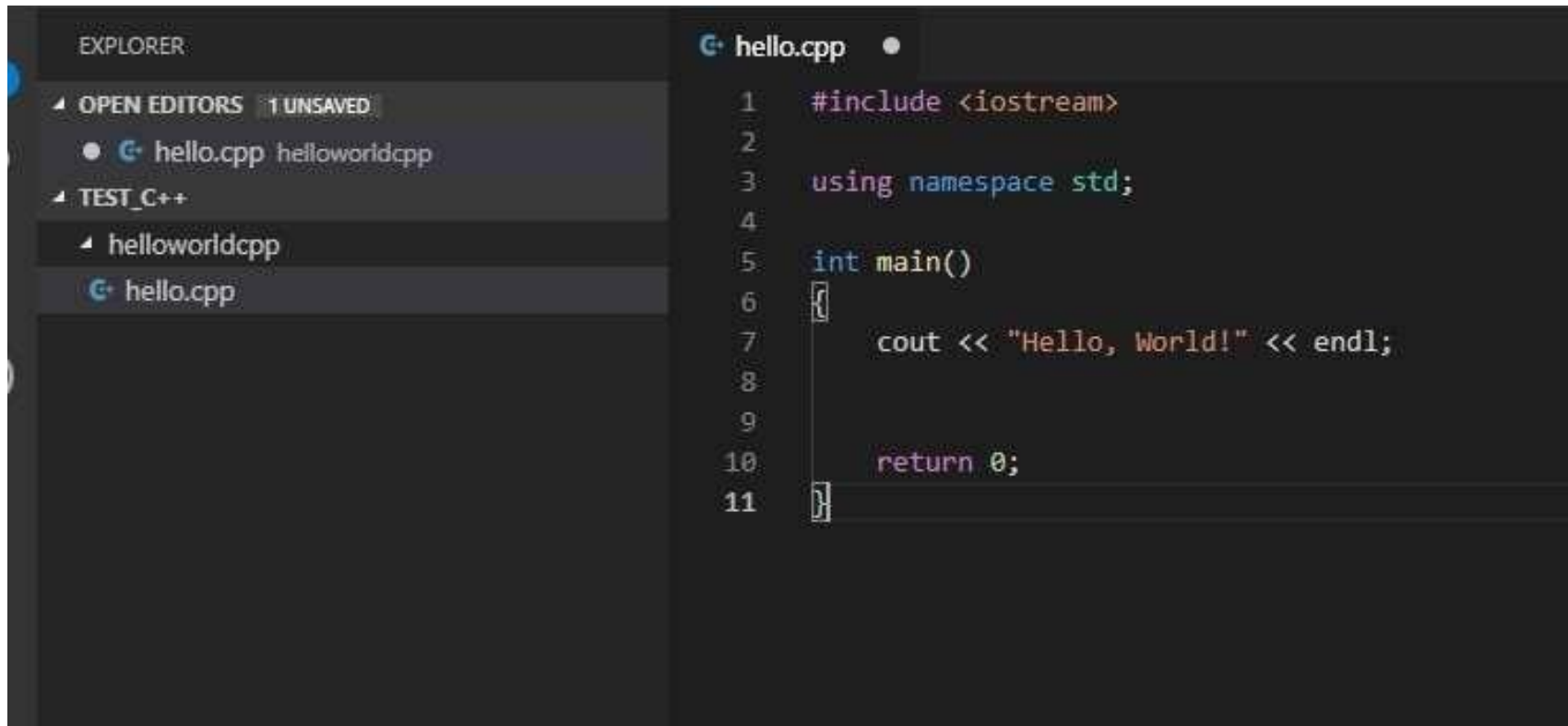
C++ Programming Environment

- Click the new file icon and create hello.cpp



C++ Programming Environment

- Enter the following code into the Hello.cpp file and press Ctrl + S to save.

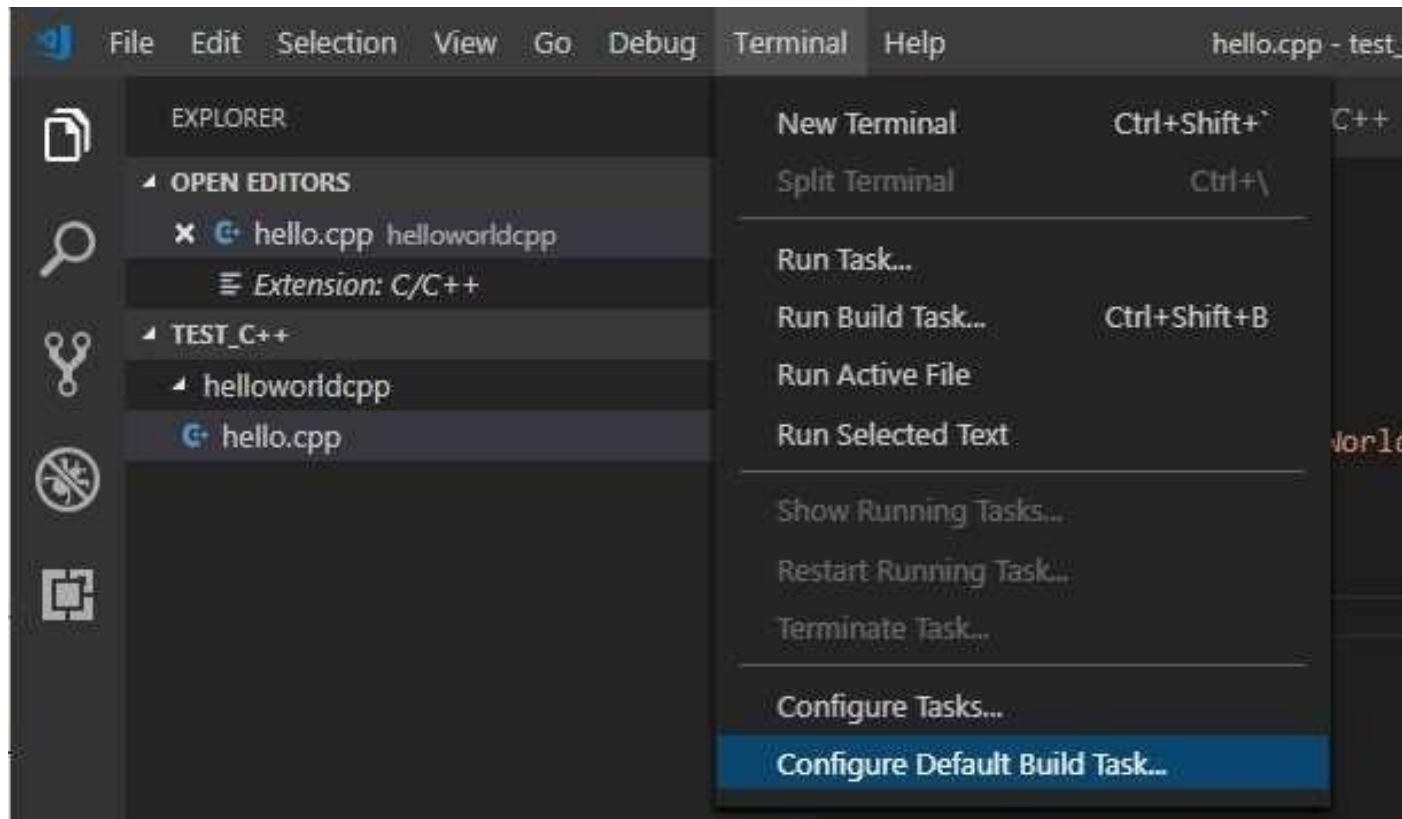


The screenshot shows a code editor interface with a dark theme. On the left, the 'EXPLORER' sidebar shows a project structure with folders 'TEST_C++' and 'helloworldcpp', and a file 'hello.cpp' selected. The main editor area displays the following C++ code in a file named 'hello.cpp':

```
1  #include <iostream>
2
3  using namespace std;
4
5  int main()
6  {
7      cout << "Hello, World!" << endl;
8
9
10     return 0;
11 }
```

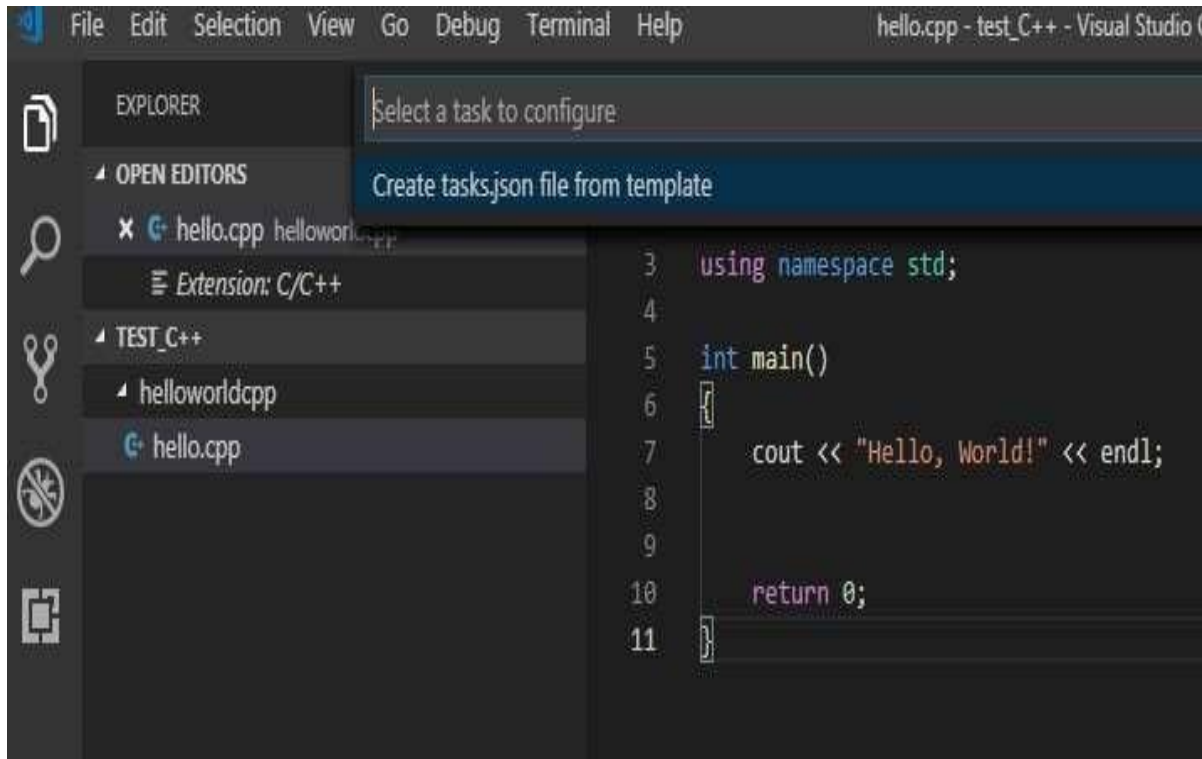
C++ Programming Environment

- From the menu of the Visual Studio Code, select Terminal > Default Build Job Configuration.

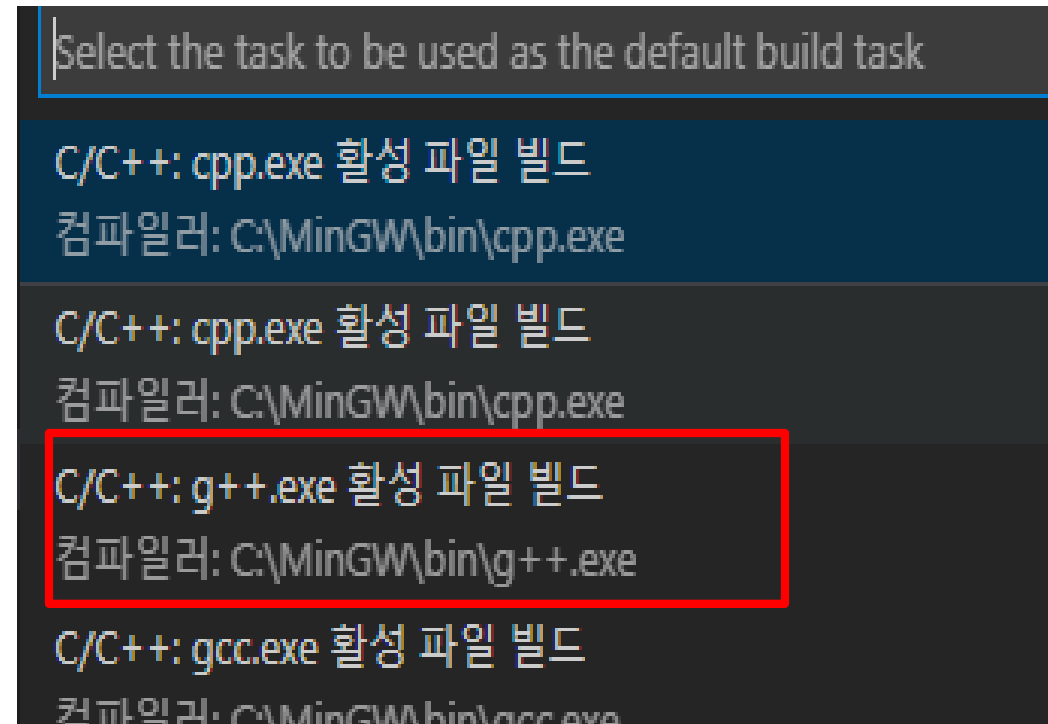


C++ Programming Environment

- Click Create tasks.json file from template.

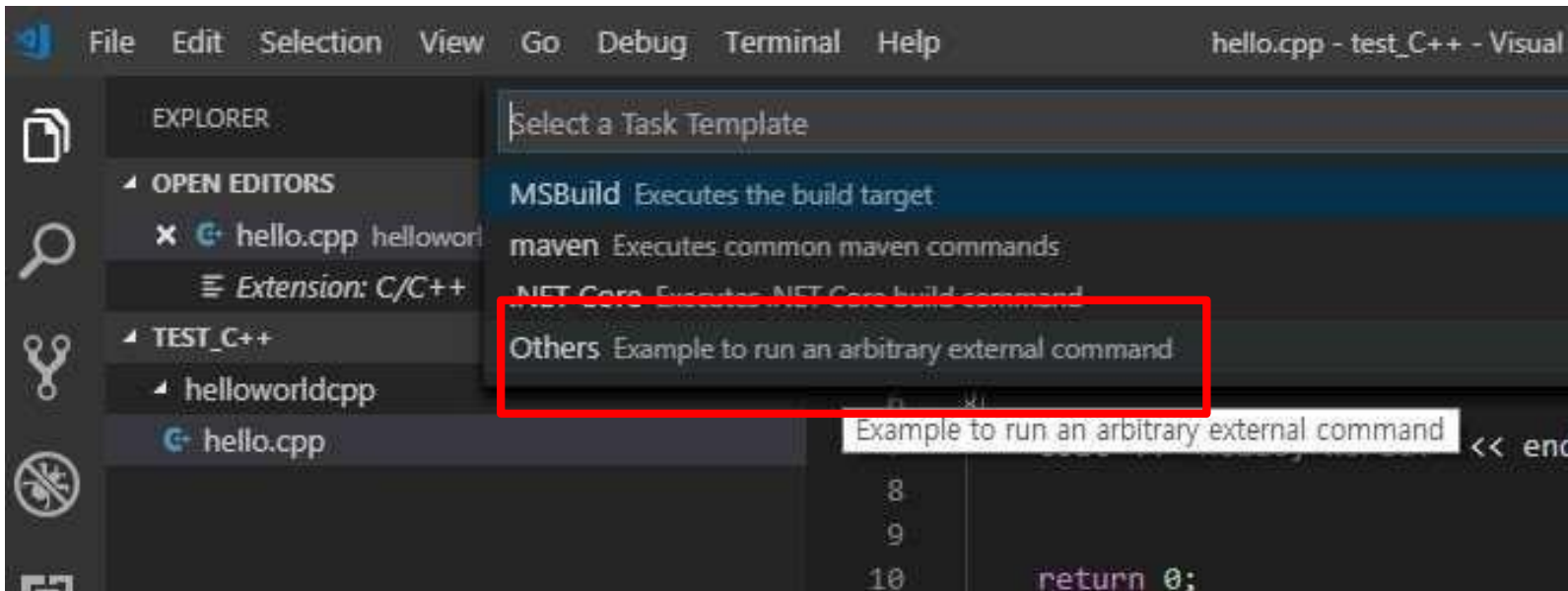


OR



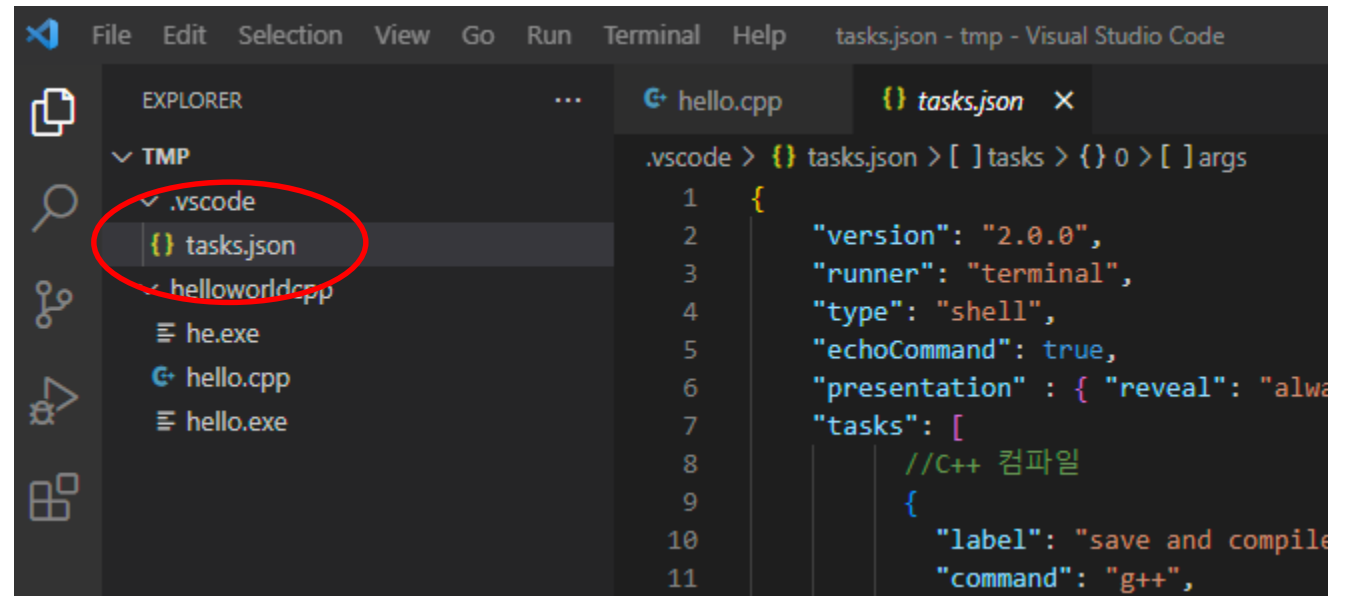
C++ Programming Environment

- Or Click Others.



C++ Programming Environment

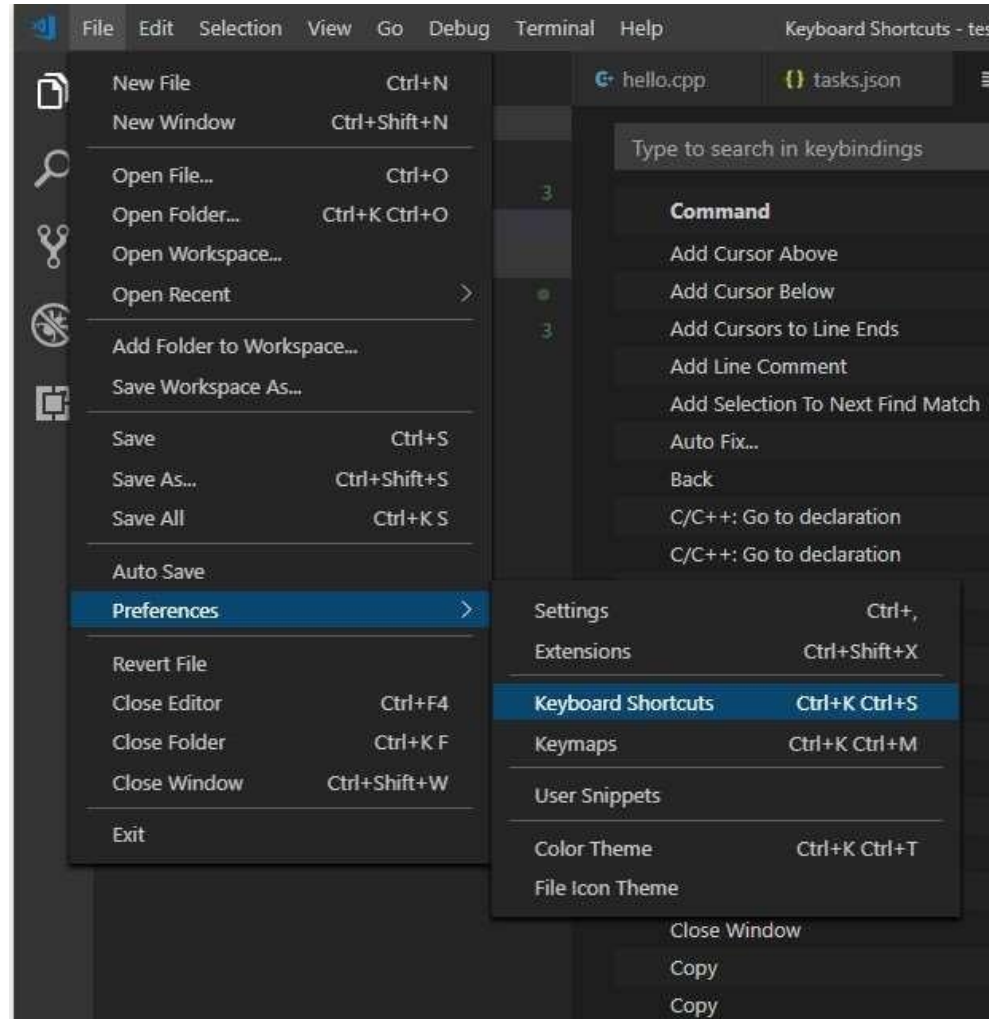
- Copy the json file uploaded on the homepage and eTL to "tasks.json"



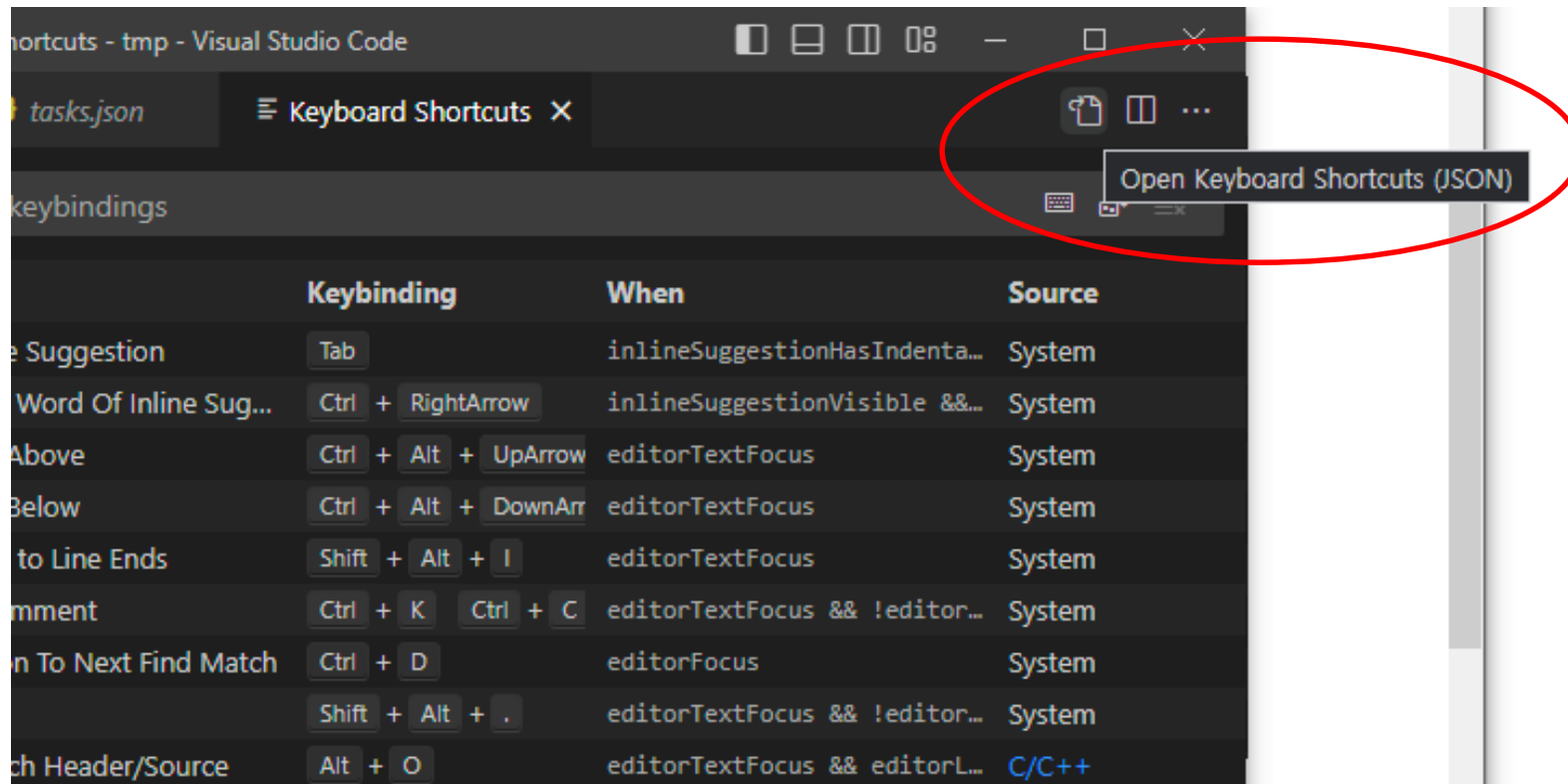
```
.vscode > {} tasks.json > [ ] tasks > {} 0 > [ ] args
1  {
2      "version": "2.0.0",
3      "runner": "terminal",
4      "type": "shell",
5      "echoCommand": true,
6      "presentation": { "reveal": "always" },
7      "tasks": [
8          //C++ 컴파일
9          {
10             "label": "save and compile",
11             "command": "g++",
```

C++ Programming Environment

- For your convenience, set the shortcut key.



C++ Programming Environment



C++ Programming Environment

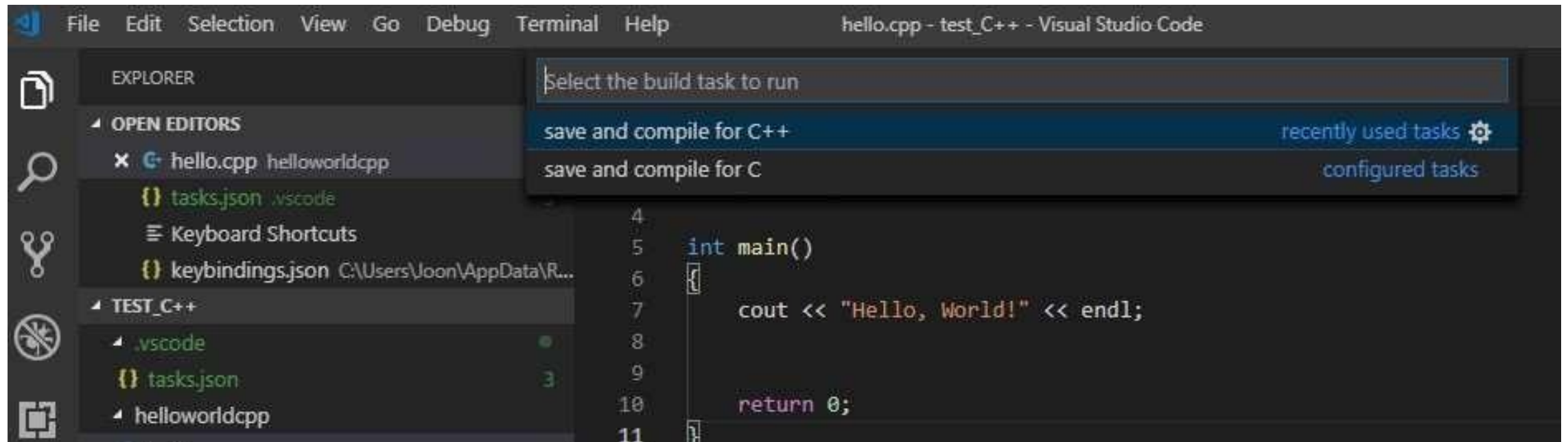
- Enter and press Ctrl + S to save as follows
(단 우분투의 경우 데스크톱 환경의 단축키가 우선입니다. 다른 곳에서 사용하지 않는 단축키를 사용해야 합니다.)

- You can use other key bindings for "build" and "run"

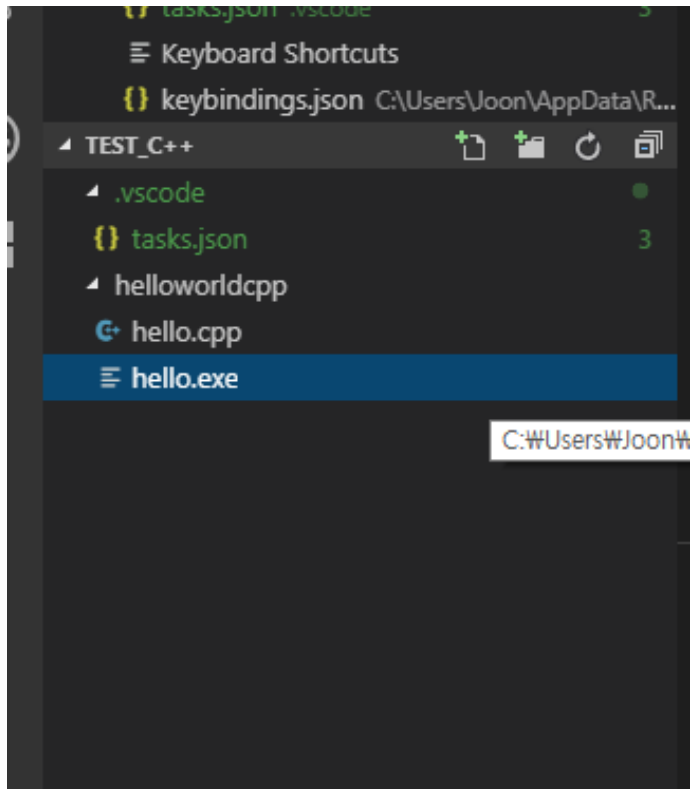
```
1 // Place your key bindings in this file to override the defaults
2 [
3     //build
4     {"key": "ctrl+b", "command": "workbench.action.tasks.build"},
5     //run
6     {"key": "ctrl+r", "command": "workbench.action.tasks.test"}
7 ]
```

C++ Programming Environment

- In Hello.cpp, press Ctrl +b and click save and compile for C++.



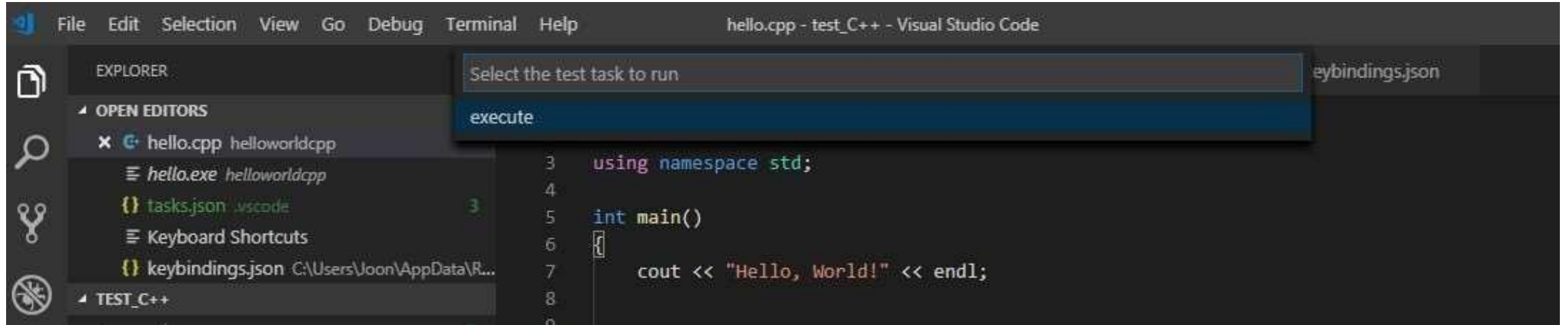
C++ Programming Environment



- All files being edited will be saved and the compilation progress will be shown in the terminal before.
- If the compile was run without any problems, the file Hello.exe, the compilation result, will be displayed in the left navigator.

C++ Programming Environment

- Press Ctrl + r and click execute
- The results of the execution are displayed in the terminal.



```
File Edit Selection View Go Debug Terminal Help hello.cpp - test_C++ - Visual Studio Code
```

EXPLORER

OPEN EDITORS

- hello.cpp helloworldcpp
- hello.exe helloworldcpp
- tasks.json vscode
- Keyboard Shortcuts
- keybindings.json C:\Users\Joon\AppData\Roaming\Code\User\keybindings.json

TEST_C++

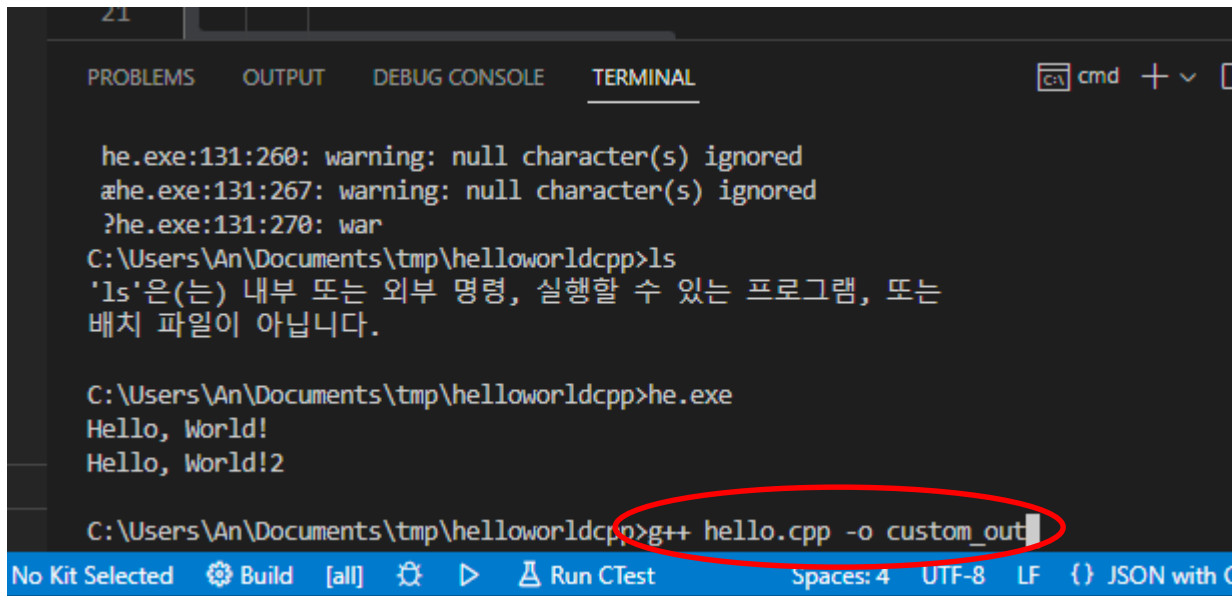
```
3 using namespace std;
4
5 int main()
6 {
7 cout << "Hello, World!" << endl;
8
9
```

```
Hello, World!

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

C++ Programming Environment

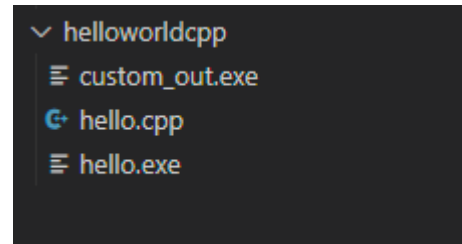
- Or you can use the terminal to compile c++ files with "g++"



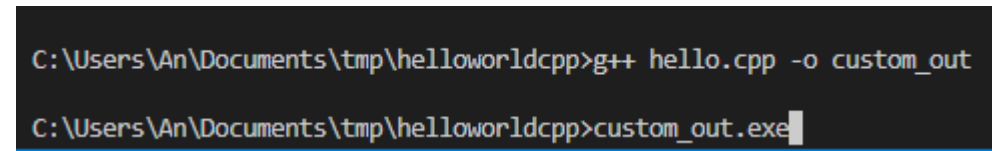
```
he.exe:131:260: warning: null character(s) ignored
æhe.exe:131:267: warning: null character(s) ignored
?he.exe:131:270: war
C:\Users\An\Documents\tmp\helloworldcpp>ls
'ls'은(는) 내부 또는 외부 명령, 실행할 수 있는 프로그램, 또는
배치 파일이 아닙니다.

C:\Users\An\Documents\tmp\helloworldcpp>he.exe
Hello, World!
Hello, World!2

C:\Users\An\Documents\tmp\helloworldcpp>g++ hello.cpp -o custom_out
```



```
helloworldcpp
├── custom_out.exe
├── hello.cpp
└── hello.exe
```



```
C:\Users\An\Documents\tmp\helloworldcpp>g++ hello.cpp -o custom_out
C:\Users\An\Documents\tmp\helloworldcpp>custom_out.exe
```

C++ Example code 1

- Try to compile this code and print out the results.
- What functions should be added to change private variable? (indirect approach)

```
1  #include <iostream>
2  #include <string>
3  using namespace std;
4
5  class Item { // Class definition
6  public:
7      string title;
8      double price;
9      double SalePrice() { return (price*0.9);}
10     bool isAvailable() { return (inStockQuantity > 0); }
11 private:
12     int inStockQuantity;
13 };
14
15 int main(void)
16 {
17     Item a;
18     a.title="comp";
19     a.price=2000;
20     cout << a.title <<endl;
21     cout << a.SalePrice() << endl;
22     return 0;
23 }
24
```

C++ Example code 2

- Try to compile this code and print out the results.

```
1  #include <iostream>
2  #include <string>
3  #include <cstring>
4  #include <assert.h>
5  using namespace std;
6
7  class String {
8  public:
9      String(const char *s) {
10         len = strlen(s);
11         str = new char[len + 1];
12         assert(str != 0);
13         strcpy(str,s);
14     }
15
16     ~String() { delete [] str; }
17     void showStr()
18     {
19         cout<<str<<endl;
20     }
21
22 private:
23     int len;
24     char *str;
25 };
26
27 int main(void)
28 {
29     String str = String("str"); // Definition
30     str.showStr();
31     return 0;
32 }
```

Java Programming Environment Setup

- Download Java JDK and set Path
- <https://codedragon.tistory.com/8510>
- <https://limkydev.tistory.com/61>
- For reference

Java Programming Environment Setup

- <https://www.jetbrains.com/idea/download/#section=windows>



Version: 2018.3.5
Build: 183.5912.21
Released: February 26, 2019
[Release notes](#)

Download IntelliJ IDEA

Windows

macOS

Linux

Ultimate

For web and enterprise development

DOWNLOAD

.EXE
▼

Community

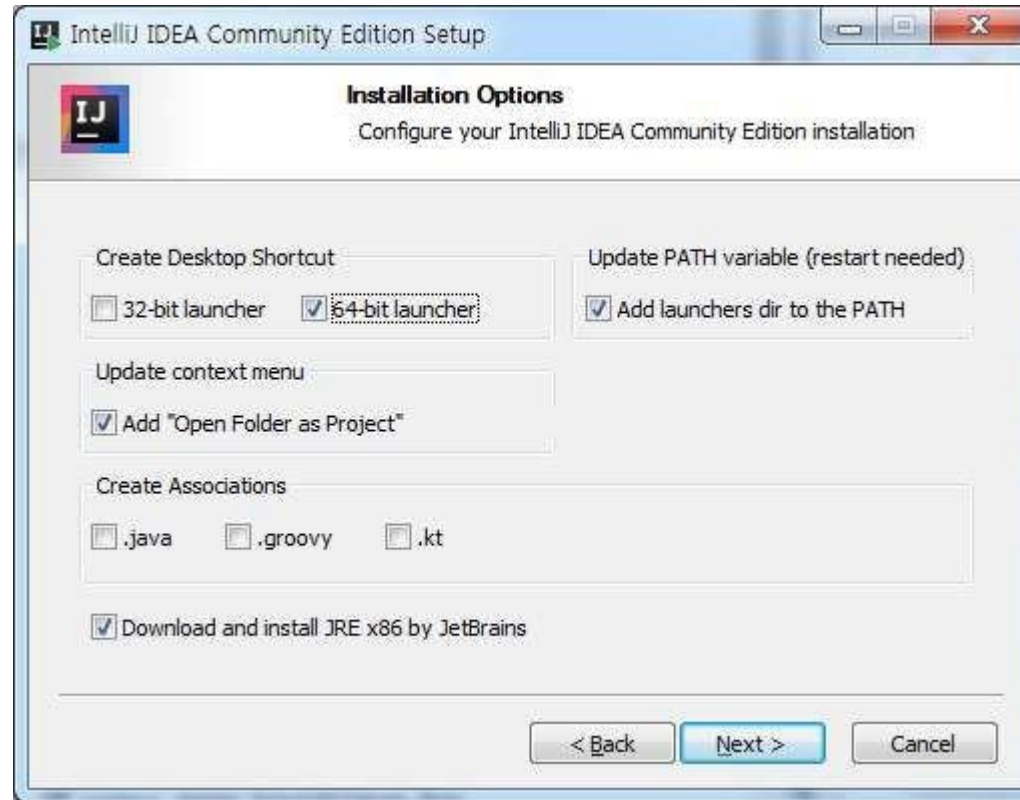
For JVM and Android development

DOWNLOAD

.EXE
▼

Installation 2 (For Windows)

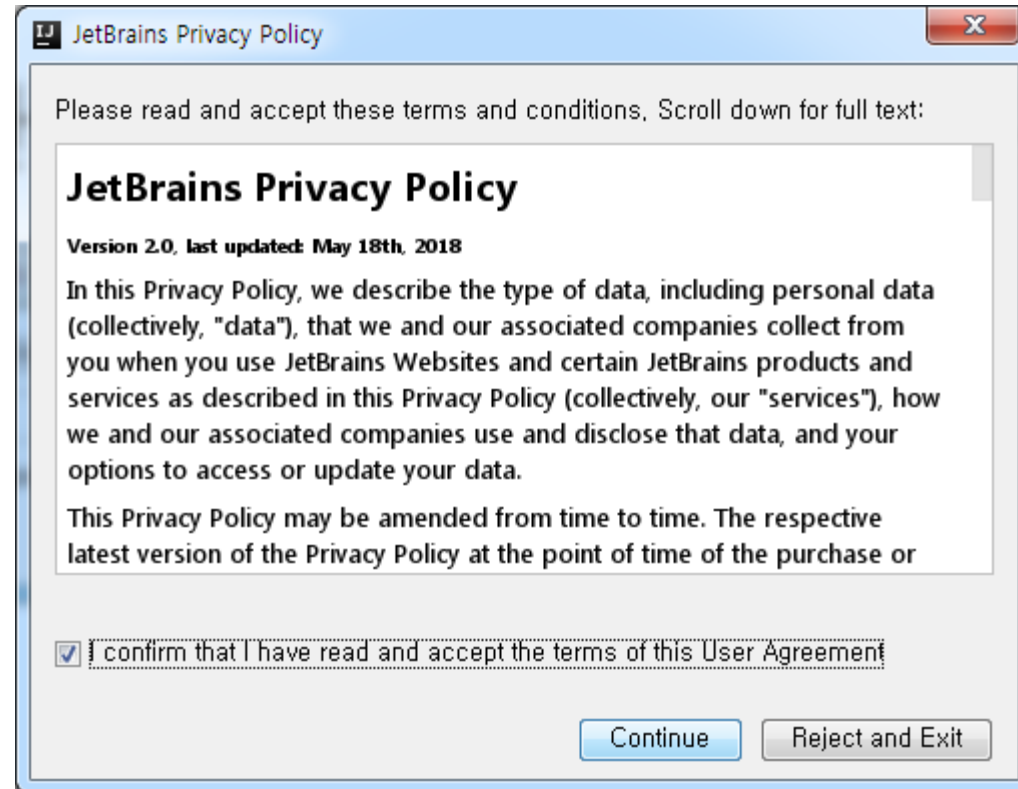
Next > ... >



→ Install

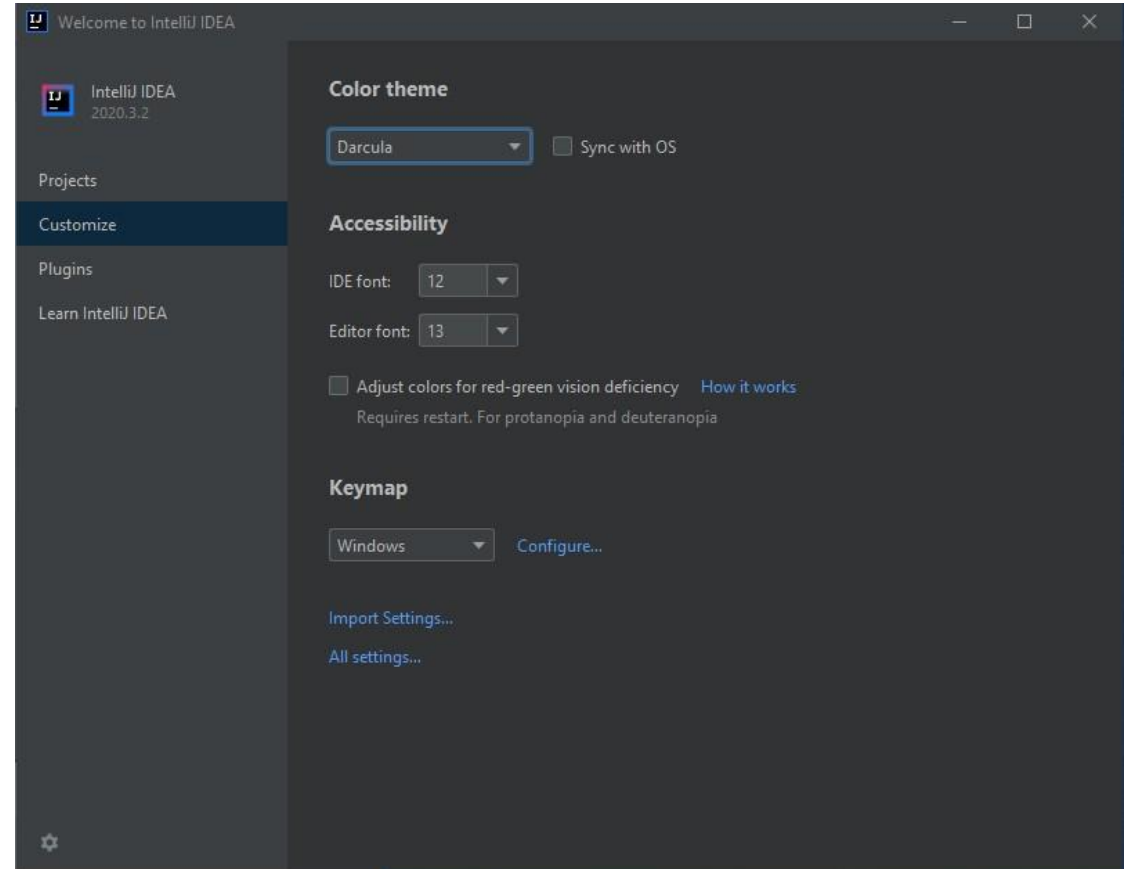
Installation 3 (For Windows)

- Reboot the computer and run program

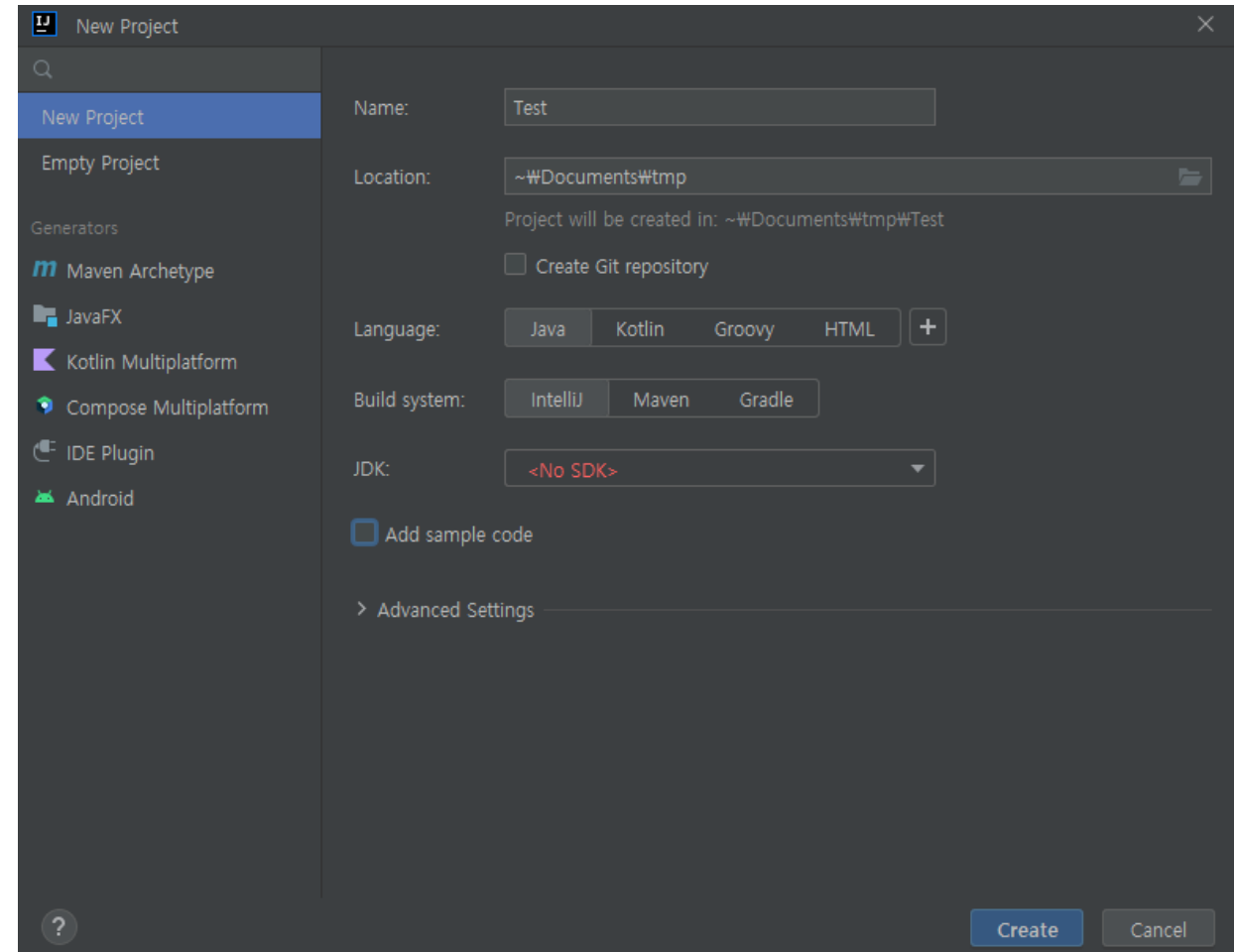
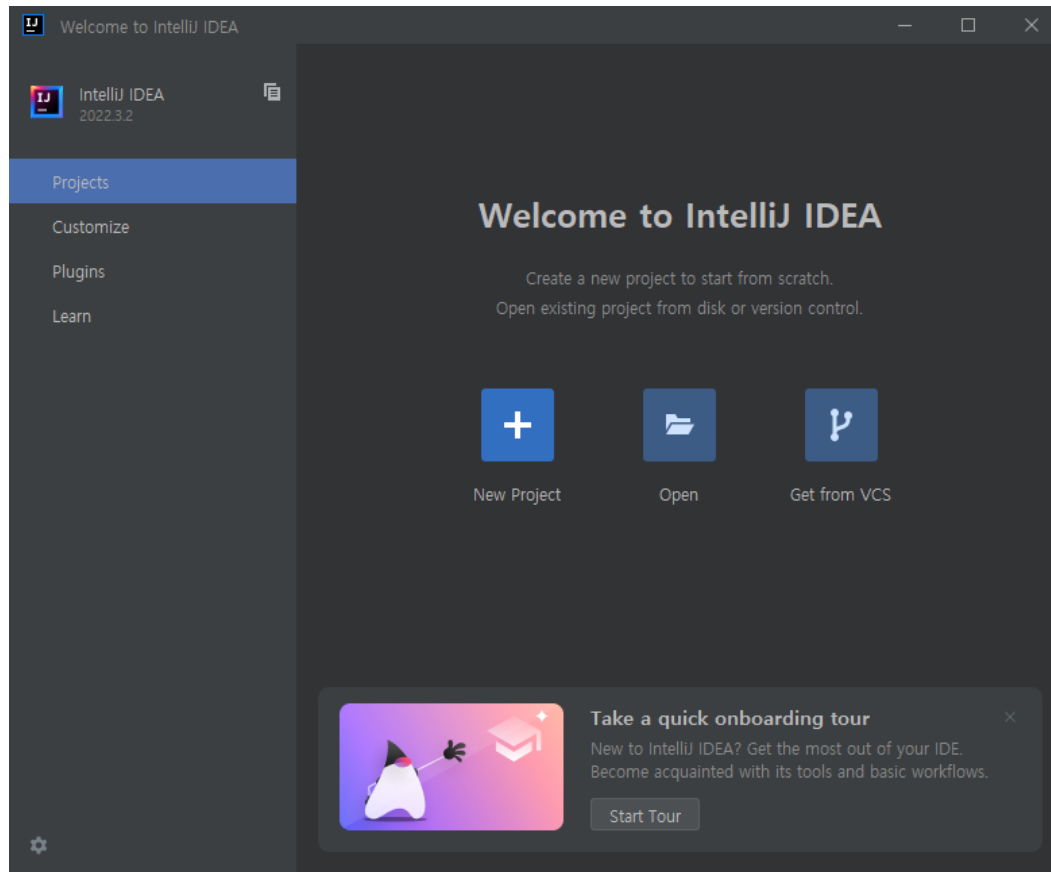


Installation 4 (For Windows)

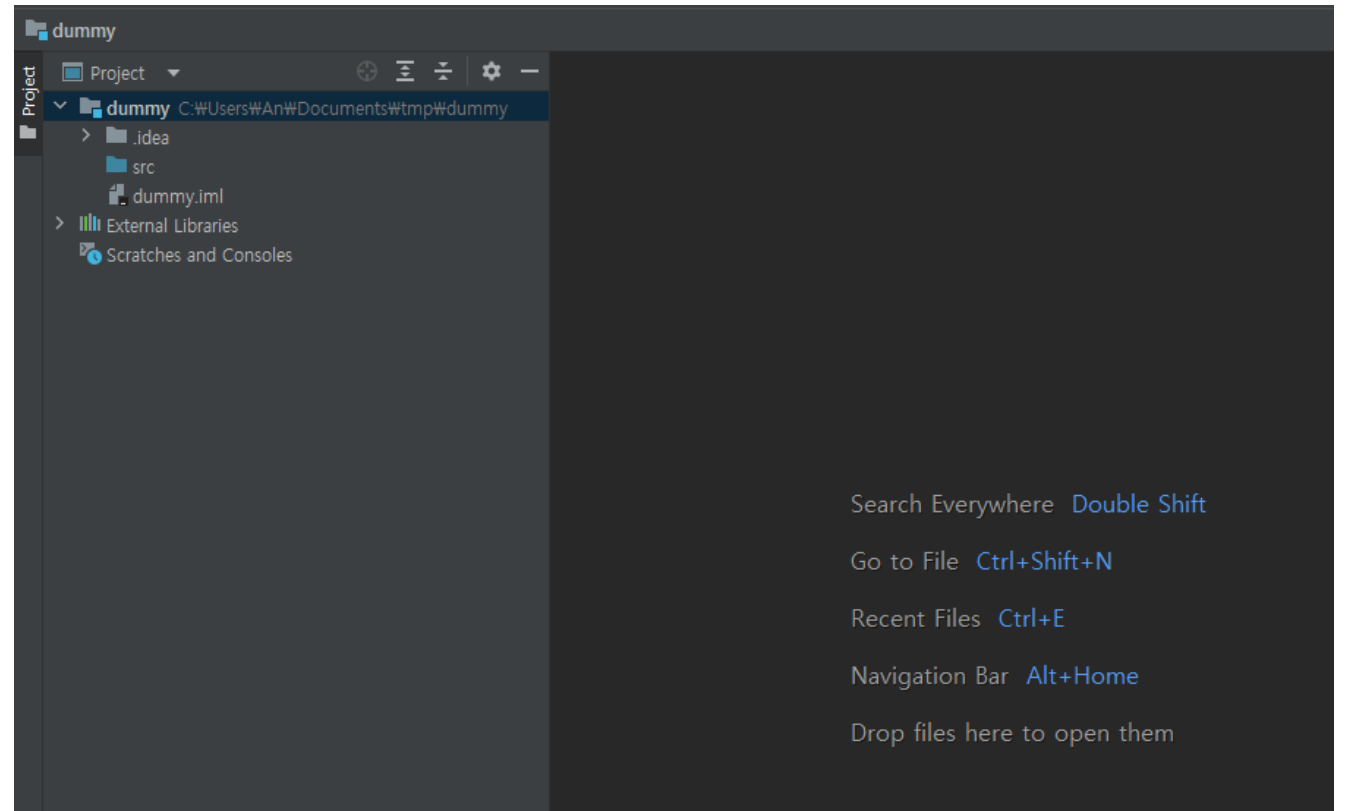
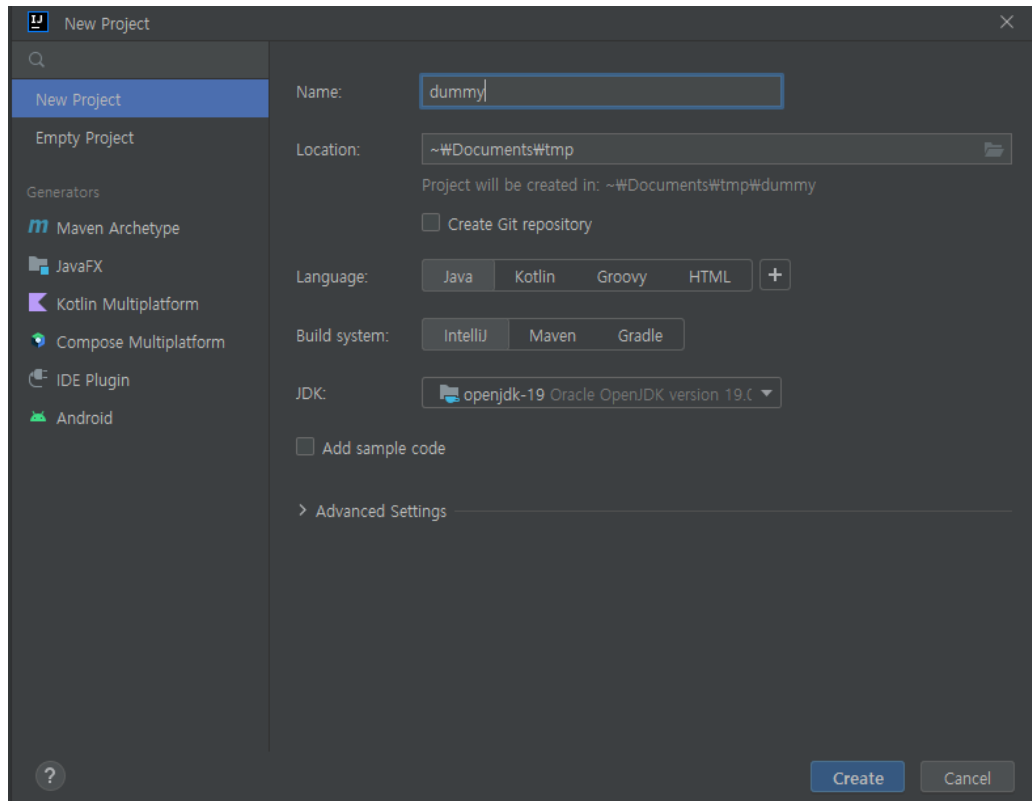
- Select either Darcula / Light
- or whatever style you want



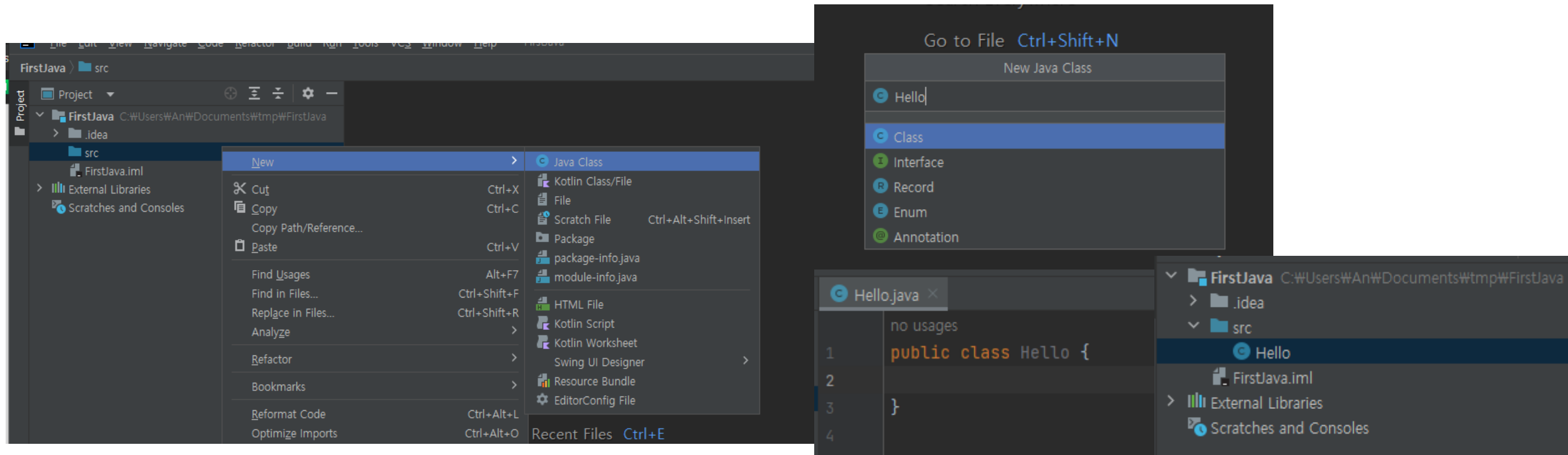
First Start-up settings



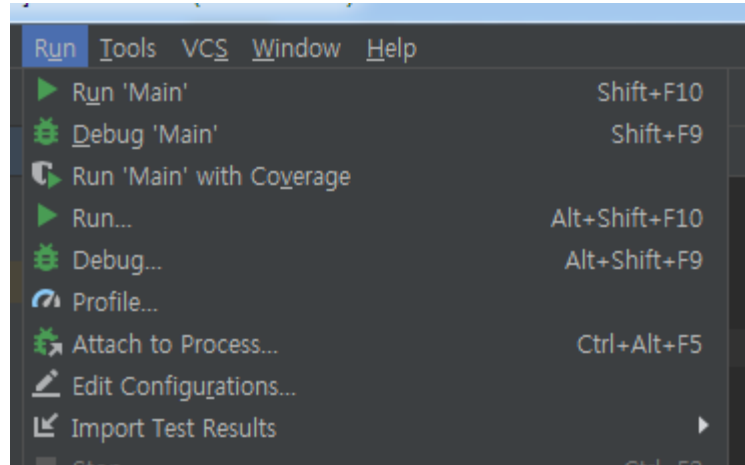
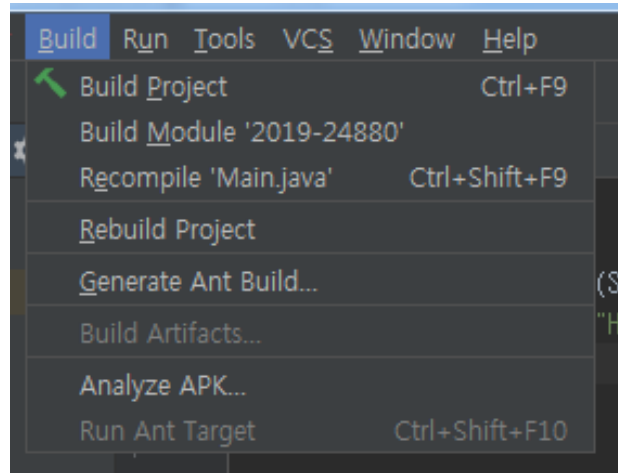
First Start-up settings



First Start-up settings



Build and run



Build (CTRL+F9)-> Run (SHIFT + F10)

Example 1

```
Hello.java x
no usages
1 ▶ public class Hello {
    no usages
2 ▶     public static void main(String[] args) {
3     System.out.println("Hello World!");
4     }
5 }
6
```

result

```
"C:\Program Files\Java\jdk-11.0.1\bin\java.exe"
Hello World!

Process finished with exit code 0
```

Example 2

```
3 ▶ public class Main {
4
5     public String title;
6     public double price;
7     private int inStockQuantity;
8     public double SalePrice() { return (price * 0.9);}
9     public boolean isAvailable(){
10         if(inStockQuantity > 0) return true;
11         else return false;
12     }
13
14 ▶ public static void main(String[] args) {
15     // write your code here
16     Main A = new Main();
17     A.title = "comp";
18     A.price = 1000;
19     System.out.println(A.SalePrice());
20 }
21 }
```

Example 3

```
Main.java x
1  import java.util.Scanner;
2
3  ▶ public class Main {
4
5  ▶   public static void main(String[] args) {
6
7       Scanner scanner = new Scanner(System.in);
8       System.out.println("Enter username : ");
9
10      String userName = scanner.nextLine();
11      System.out.println("Username is: " + userName);
12  }
13 }
```


Example 3

Input Types

In the example above, we used the `nextLine()` method, which is used to read Strings. To read other types, look at the table below:

| Method | Description |
|----------------------------|--|
| <code>nextBoolean()</code> | Reads a <code>boolean</code> value from the user |
| <code>nextByte()</code> | Reads a <code>byte</code> value from the user |
| <code>nextDouble()</code> | Reads a <code>double</code> value from the user |
| <code>nextFloat()</code> | Reads a <code>float</code> value from the user |
| <code>nextInt()</code> | Reads a <code>int</code> value from the user |
| <code>nextLine()</code> | Reads a <code>String</code> value from the user |
| <code>nextLong()</code> | Reads a <code>long</code> value from the user |
| <code>nextShort()</code> | Reads a <code>short</code> value from the user |

Example 4

```
Main.java x
1  import java.util.Scanner;
2
3  ▶ public class Main {
4
5  ▶  ◯ public static void main(String[] args) {
6
7      Scanner scanner = new Scanner(System.in);
8      System.out.println("Enter username : ");
9      String userName = scanner.nextLine();
10
11     System.out.println("Enter age :");
12     int age = scanner.nextInt();
13
14
15     System.out.println("Username is : " + userName + ", and age is : "+age);
16     ◯ }
17 }
```