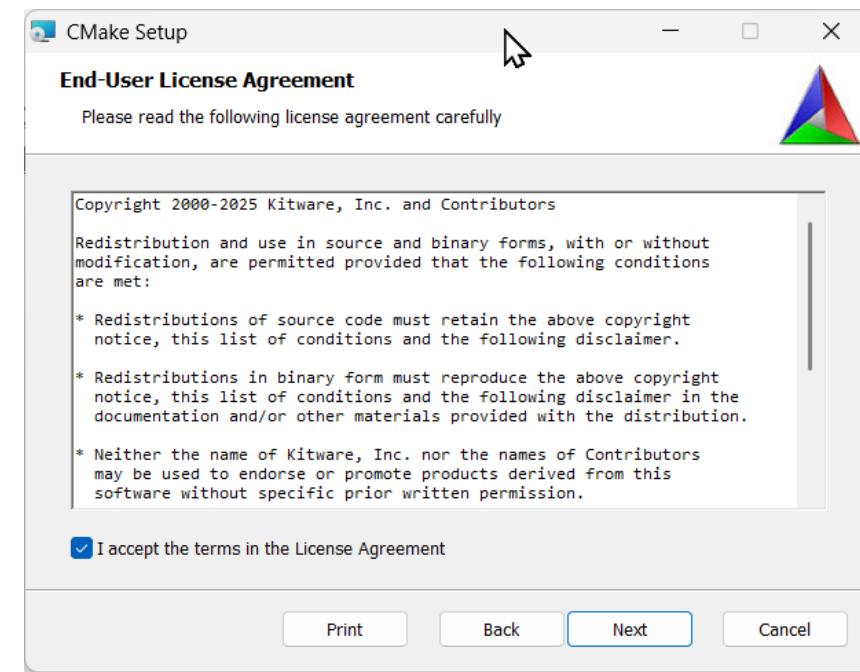
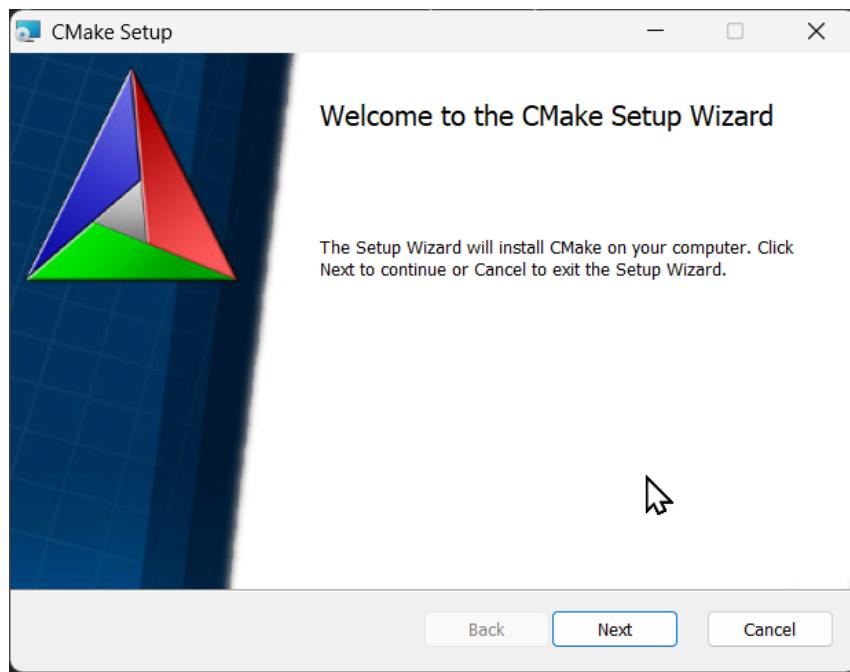
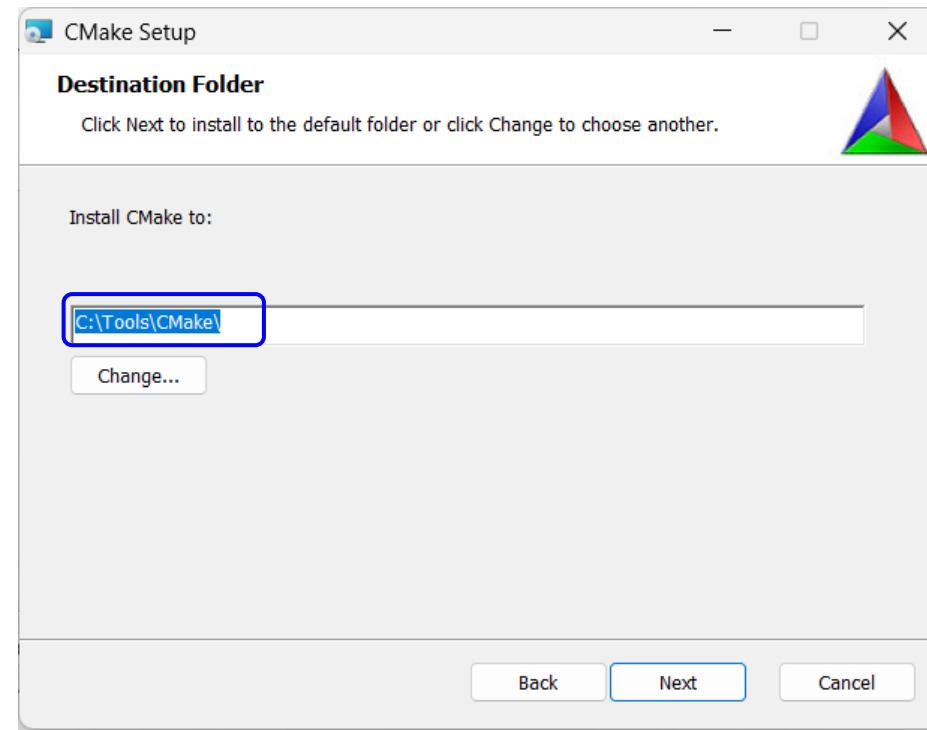
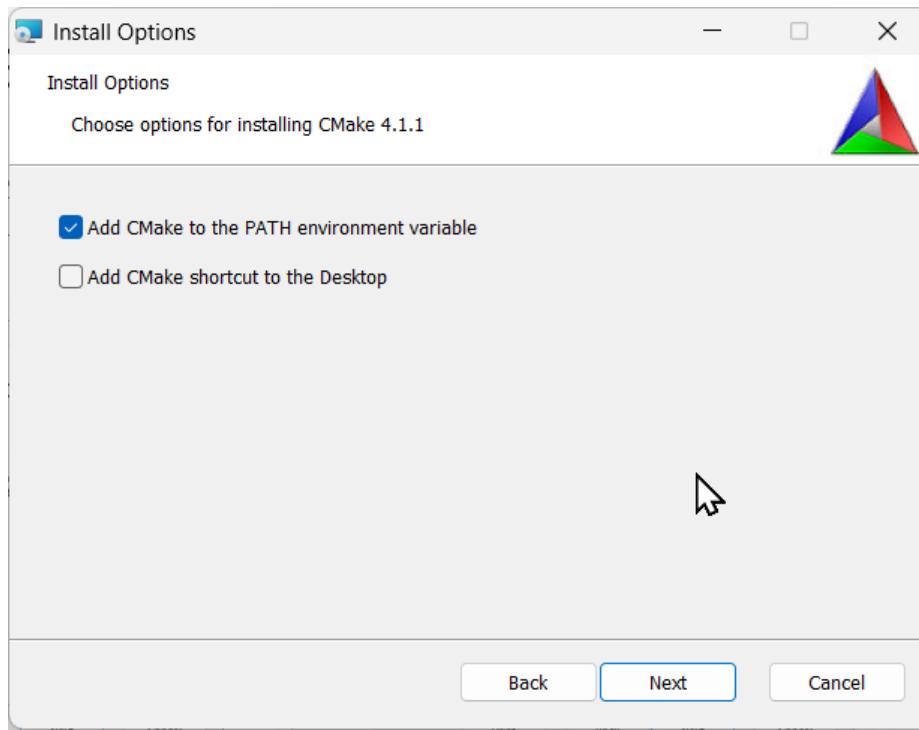


# Step1: CMake Installation (Windows)

[https://github.com/Kitware/CMake/releases/download/v4.1.2/cmake-4.1.2-windows-x86\\_64.msi](https://github.com/Kitware/CMake/releases/download/v4.1.2/cmake-4.1.2-windows-x86_64.msi)





コマンドラインで使うソフトウェアは、スペースを含まない英語のディレクトリにインストールするのがベストです。不必要なトラブルを避けられます。

```
C:\Users\sxwin>cmake --version
cmake version 4.1.2

CMake suite maintained and supported by Kitware (kitware.com/cmake).
```

「cmake --version」をコマンドラインで実行してインストールの成功を確認する。

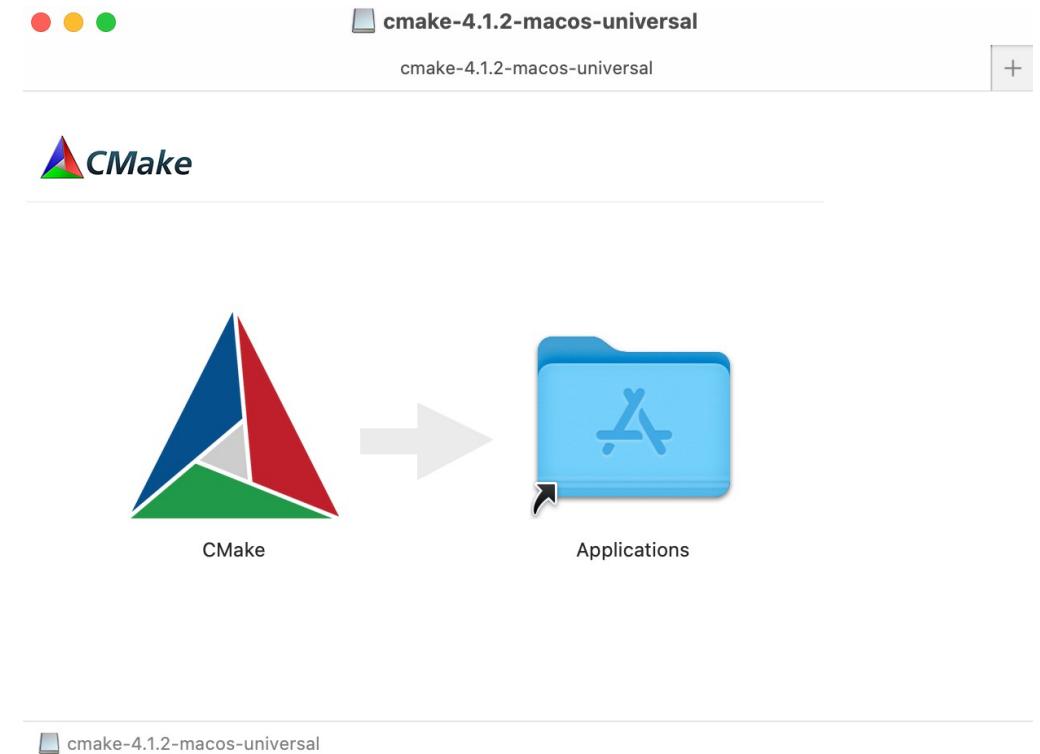
# Step1: CMake Installation (Mac)

macOS 10.13 or later

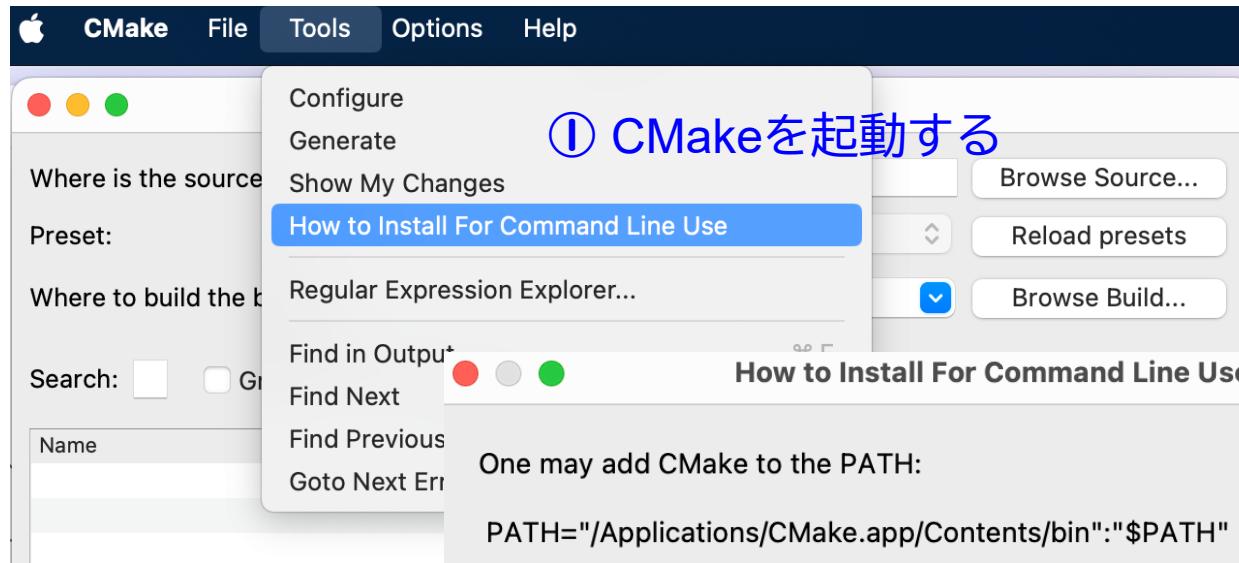
<https://github.com/Kitware/CMake/releases/download/v4.1.2/cmake-4.1.2-macos-universal.dmg>

macOS 10.10 or later

<https://github.com/Kitware/CMake/releases/download/v4.1.2/cmake-4.1.2-macos10.10-universal.dmg>



**Download & install**



① CMakeを起動する

### How to Install For Command Line Use

One may add CMake to the PATH:

```
PATH="/Applications/CMake.app/Contents/bin":$PATH"
```

Or, to install symlinks to '/usr/local/bin', run:

```
sudo "/Applications/CMake.app/Contents/bin/cmake-gui" --install
```

Or, to install symlinks to another directory, run:

```
sudo "/Applications/CMake.app/Contents/bin/cmake-gui" --install=/path/to/bin
```

② Copy it

③ ターミナルでこのコマンド  
を実行する

```
~ % sudo "/Applications/CMake.app/Contents/bin/cmake-gui" --install  
Password:  
Linked: '/usr/local/bin/cmake' -> '/Applications/CMake.app/Contents/bin/cmake'  
Linked: '/usr/local/bin/ctest' -> '/Applications/CMake.app/Contents/bin/ctest'  
Linked: '/usr/local/bin/cpack' -> '/Applications/CMake.app/Contents/bin/cpack'  
Linked: '/usr/local/bin/cmake-gui' -> '/Applications/CMake.app/Contents/bin/cmak
```

```
~ % cmake --version
```

④ 確認  
「cmake --version」

CMake suite maintained and supported by Kitware ([kitware.com/cmake](http://kitware.com/cmake)).

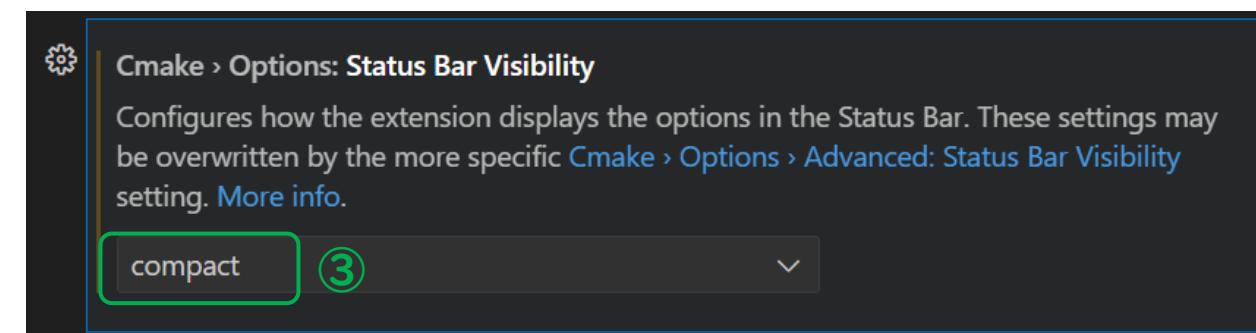
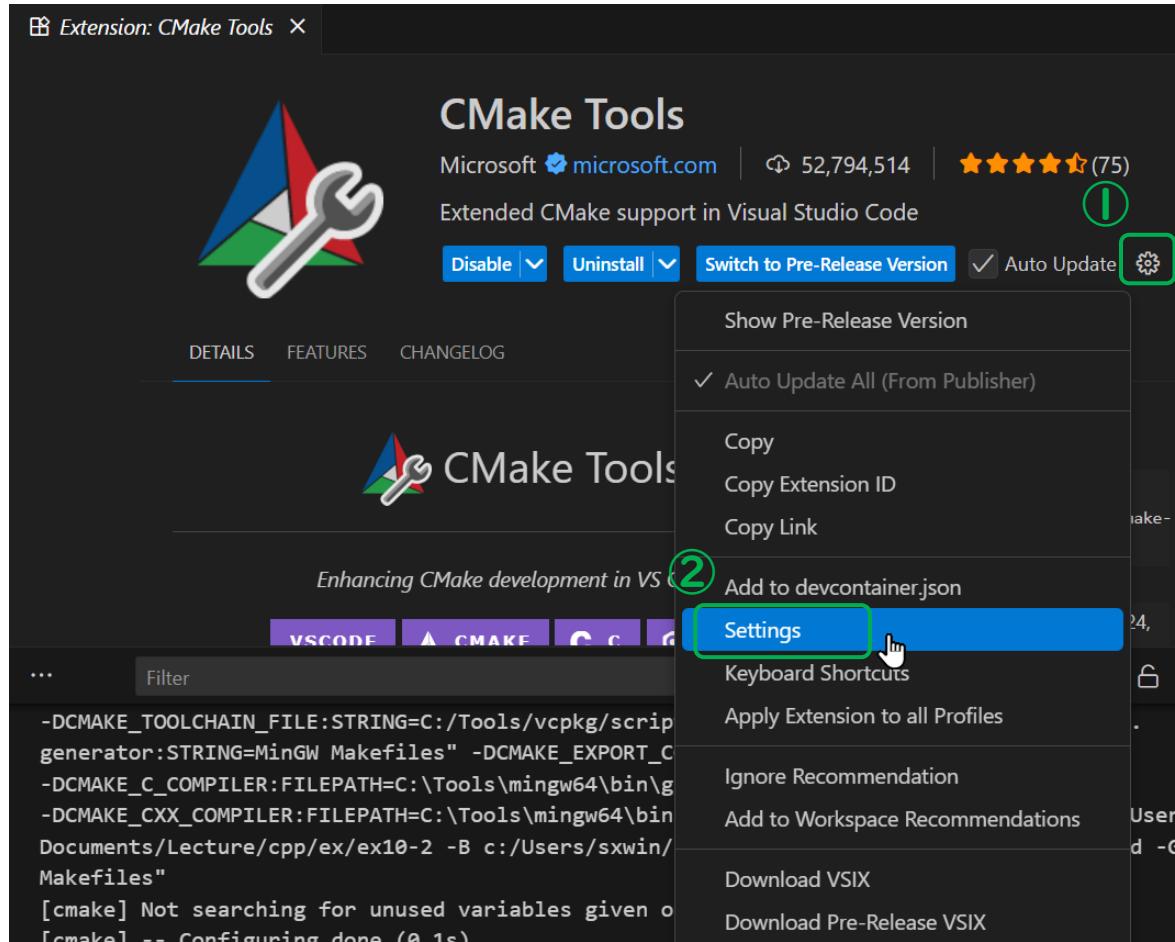
# Step2: Setup CMake in VSCode

The screenshot shows the Visual Studio Code (VSCode) Marketplace interface. The search bar at the top contains the text "cpp". The left sidebar has icons for extensions, settings, and marketplace. The main area displays search results for "cmake".

Extension	Downloads	Rating	Description	Action
CMake	45.4M	3	CMake language support for Visual Studio	<a href="#">Install</a>
<b>CMake Tools</b>	52.7M	4.5	Extended CMake support in Visual Studio	<a href="#">Install</a>
CMake Language Sup...	1.6M	4	CMake language support for VS Code	<a href="#">Install</a>
cmake-format	471K	4	Format listfiles so they don't look like...	<a href="#">Install</a>

The "CMake Tools" extension by Microsoft is highlighted with a green border around its "Install" button. The right side of the screen shows the details for the "CMake Tools" extension, including its logo (a wrench and a triangle), name, developer (Microsoft), download count (52,7), rating (4.5), and a note that it is recommended based on recently opened files. It also shows an "Install" button, an "Auto Update" checkbox, and a "Marketplace" section with the identifier "ms-vscode.cmake-tools".

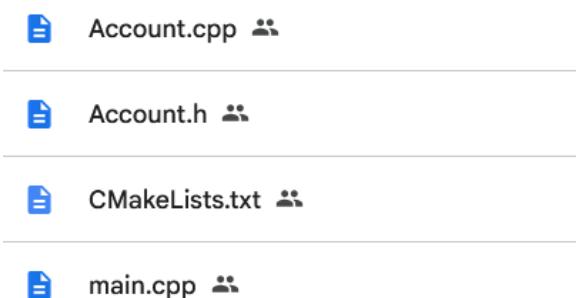
# Step2: Setup CMake in VSCode



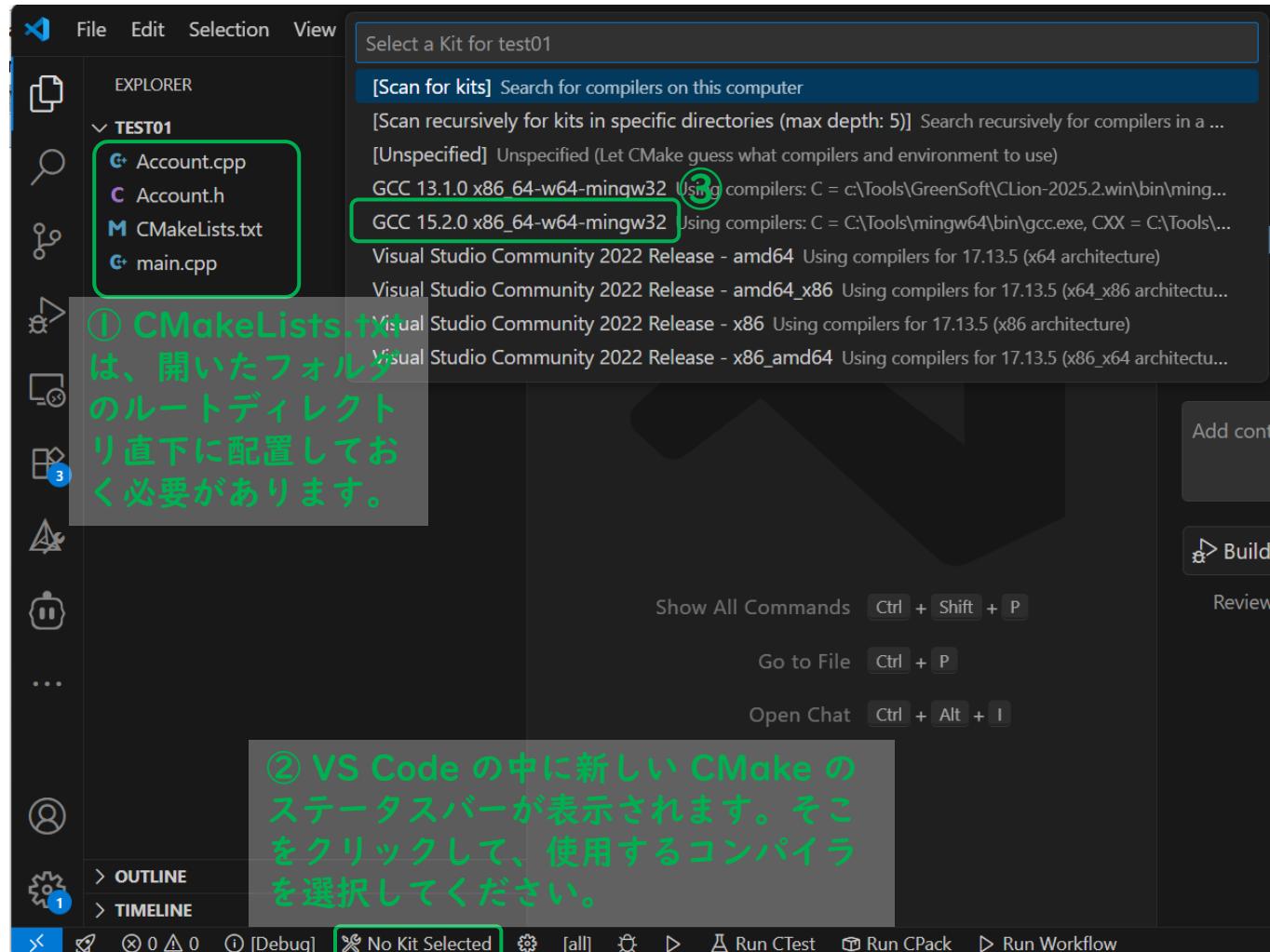
# Step3: Setup Project

1. 作業用のフォルダを新しく作成する。
2. リンク先にある4つのファイルを、そのフォルダの中にコピーする。

<https://drive.google.com/drive/folders/1G3ieFaqYydFYpbQd3qYYZNWqwU5rntYw?usp=sharing>



3. VS Code で「File ⇒ Open Folder」を選び、作成したフォルダを開く。



Account.cpp

Account.h

CMakeLists.txt

main.cpp

```
2 #include <iostream>
3 #include "Account.h"
4
5 using namespace std;
6
7 int main()
8 {
9     Account suzuki("鈴木龍一", "12345678", 1000); // 鈴木君の口座
10    Account takeda("武田浩文", "87654321", 200); // 武田君の口座
11
12    suzuki.withdraw(200); // 鈴木君が200円おろす
13    takeda.deposit(100); // 武田君が100円預ける
14
15    cout << "■鈴木君の口座：" << suzuki.name() << "(" << suzuki.no()
16    | << ")" " << suzuki.balance() << "円\n";
17
18    cout << "■武田君の口座：" << takeda.name() << "(" << takeda.no()
```

PROBLEMS OUTPUT TERMINAL ...

CMake/Launch - main - build + ⚖️ ⚖️ ⚖️ ⚖️ ⚖️

PS C:\Users\sxwin\Documents\Lecture\cpp\temp\test01\build>

ure/cpp/temp/test01/build/main.exe"
■鈴木君の口座："鈴木龍一" (12345678) 800円
■武田君の口座："武田浩文" (87654321) 300円

PS C:\Users\sxwin\Documents\Lecture\cpp\temp\test01\build>

> OUTLINE

> TIMELINE



0

△ 0

① [Debug]

✖ [GCC 15.2.0 x86\_64-w...]



[all]



[main]



▷ Run Workflow



Go Live



Win32



ここをクリックするとコードを実行できます。問題がなければ、コンパイルとリンクが正常に完了したあと、実行ファイルが自動的に起動します。

# CMakeLists.txt とは？

- CMake が利用する設定用スクリプトファイルで、ファイル名は必ず CMakeLists.txt にします。
- このファイルの中で、
  - プロジェクト名は何か
  - どの言語を使うか (C++ など)
  - どのソースファイルをビルド対象にするか
  - 実行ファイルやライブラリなど、どんな成果物を作るかといった情報を指定します。
- VS Code や CLion、あるいはコマンドラインから cmake コマンドを実行したときは、この CMakeLists.txt を読み取って、実際にコンパイル・リンクを行うためのビルド用ファイルを自動生成しています。
- 実際の大規模 C++ プロジェクトでは、CMakeLists.txt には外部ライブラリやヘッダのパス、プリプロセッサマクロ、コンパイル／最適化オプションなど、多くのビルド設定が追加されます。こうした設定によって、CMake はプラットフォームごとに適切なコンパイルコマンドを自動生成し、複雑な依存関係を一元的に管理できるようになります。

# CMakeLists.txt の基本

- ・ シンプルな C++ プロジェクトであれば、右側の CMakeLists.txt テンプレートをそのまま利用できます。
  - ・ (1) の欄にプロジェクト名（英数字のみ・スペース不可）を入力
  - ・ (2) の欄に生成する実行ファイル名を入力
  - ・ (3) の欄にコンパイルしてリンクしたい複数のソースファイル名を入力
- ・ CMake のより複雑な使い方については、こちらを参照してください。
  - ・ [C++初心者のためのCMake基本構文と使い方を徹底解説](#)
  - ・ [【超入門】1日で理解するCmake](#)
- ・もちろん、AIに自分の要件をしっかり伝えてしまえば、CMakeLists.txtを直接生成してもらうのが、実は効率的な方法です。

```
# 1. このプロジェクトに必要な最低限の CMake  
バージョンを指定します  
cmake_minimum_required(VERSION 3.10)
```

```
# 2. プロジェクト名を定義します。  
# 'CXX' は C++ 言語を使用することを示します。  
project(myapp_CXX)
```

```
# 3. C++ 標準バージョンを指定 (例: C++17)  
set(CMAKE_CXX_STANDARD 17)  
set(CMAKE_CXX_STANDARD_REQUIRED True)
```

```
# 4. 実行可能ターゲットの定義  
# add_executable(目標名 ソースファイルのリスト)  
# CMake は依存関係とリンクを自動的に処理します  
add_executable(  
    main  
    main.cpp②  
    Account.cpp③  
)
```



```
g++ -std=c++17 main.cpp Account.cpp -o main
```