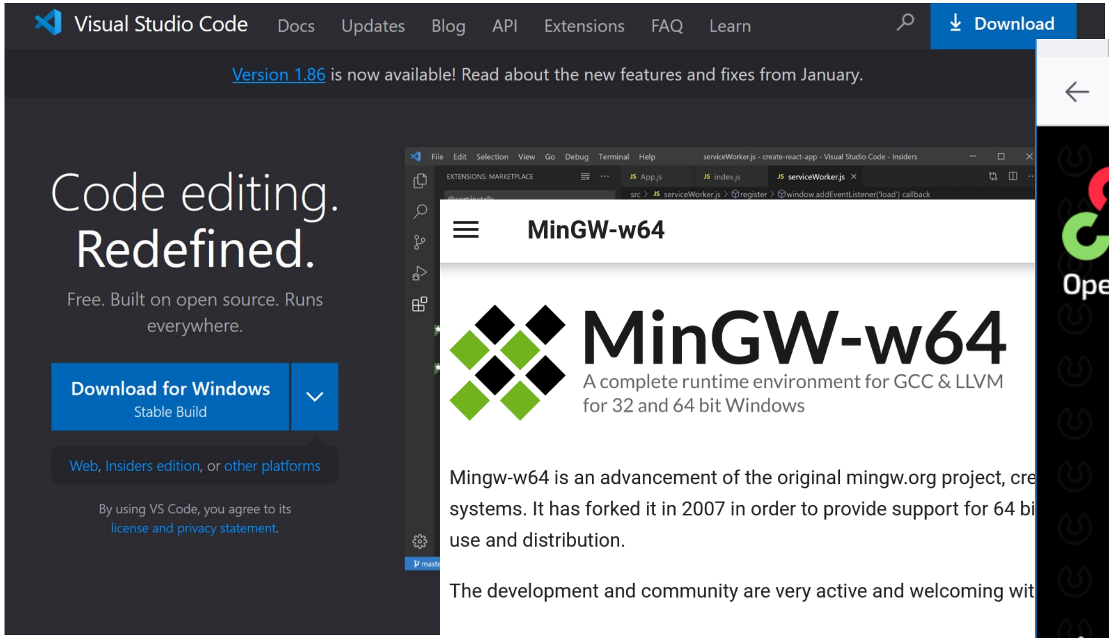
Install Visual Studio Code, gcc/g++ Compiler Suite and OpenCV



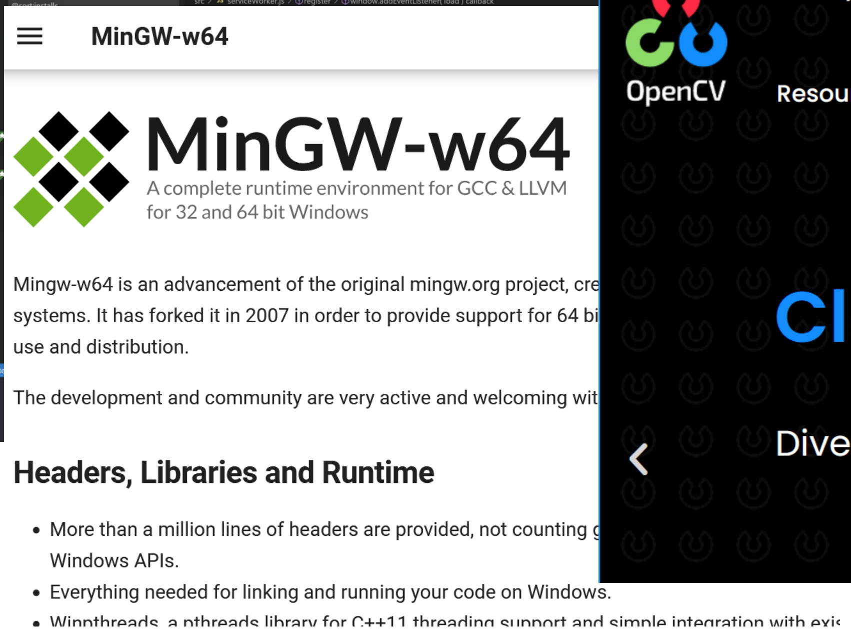
Version 1.86 is now available! Read about the new features and fixes from January.

Code editing.
Redefined.

Free. Built on open source. Runs everywhere.

Web, Insiders edition, or other platforms

By using VS Code, you agree to its license and privacy statement.



MinGW-w64

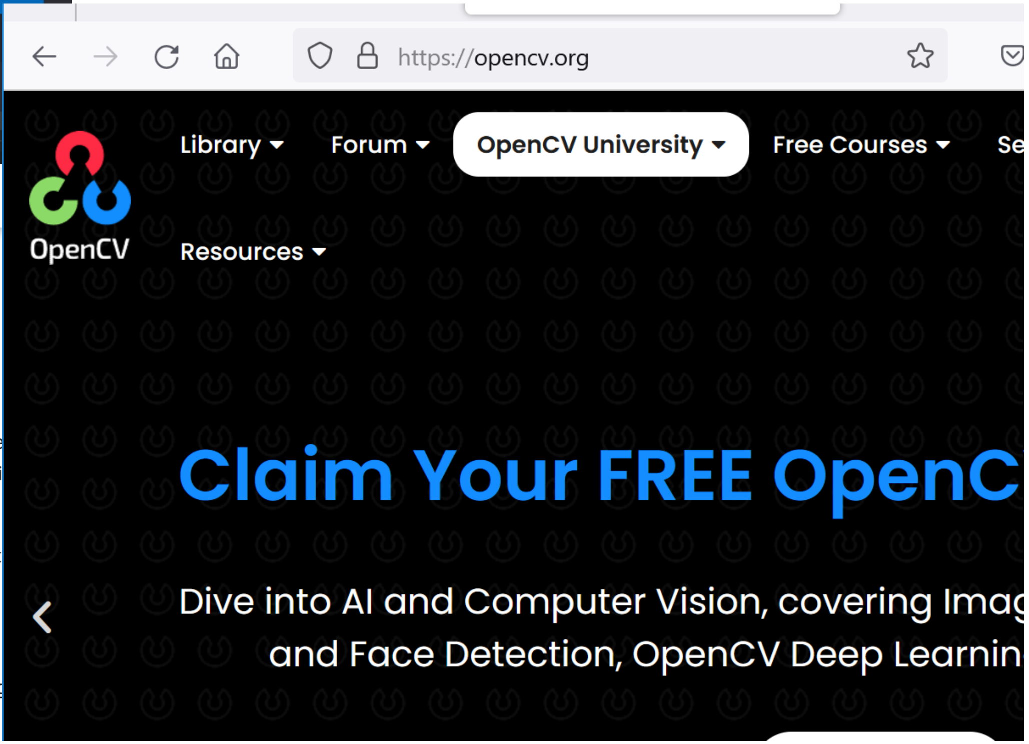
A complete runtime environment for GCC & LLVM for 32 and 64 bit Windows

Mingw-w64 is an advancement of the original mingw.org project, cre
systems. It has forked it in 2007 in order to provide support for 64 bi
use and distribution.

The development and community are very active and welcoming wit

Headers, Libraries and Runtime

- More than a million lines of headers are provided, not counting g
Windows APIs.
- Everything needed for linking and running your code on Windows.
- Winpthreads, a pthreads library for C++11 threading support and simple integration with exis

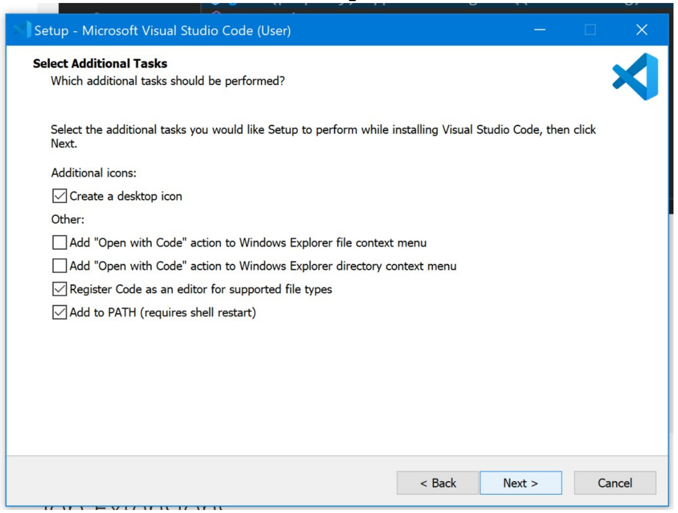


Claim Your FREE OpenC

Dive into AI and Computer Vision, covering Imag
and Face Detection, OpenCV Deep Learnin

Install Visual Studio Code

- Note: VSCode has a lot of configuration options and additional features that I found to be quite daunting at the beginning.
- Installation instructions and the download link for the VSCode Windows Installer are here: https://code.visualstudio.com/docs/setup/windows
- I chose the default options offered by the installer:



Setup - Microsoft Visual Studio Code (User)

Select Additional Tasks
Which additional tasks should be performed?

Select the additional tasks you would like Setup to perform while installing Visual Studio Code, then click Next.

Additional icons:
- [✓] Create a desktop icon

Other:
- [ ] Add "Open with Code" action to Windows Explorer file context menu
- [ ] Add "Open with Code" action to Windows Explorer directory context menu
- [✓] Register Code as an editor for supported file types
- [✓] Add to PATH (requires shell restart)

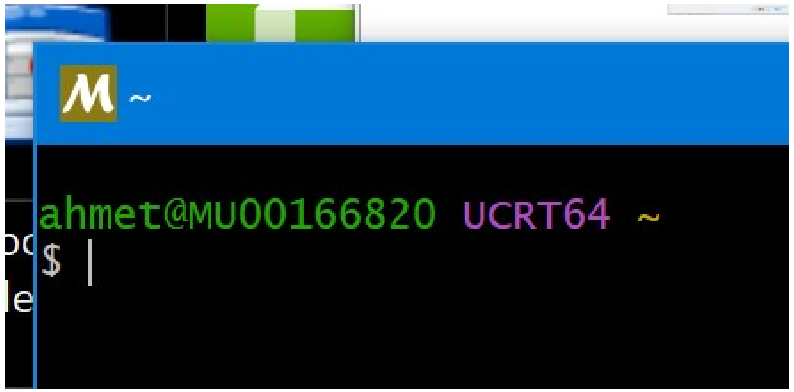
Install and Test the C/C++ Compiler Suite

- Install the C/C++ add-on and MinGW64 C/C++ compiler[*]:
  - https://code.visualstudio.com/docs/cpp/config-mingw
  - You will see the “direct link to the installer” on this page
  - I chose the default settings as suggested by the installer, so the entire MinGW64 components go under the directory C:\MSYS64
- Test your C/C++ installation as suggested by the installation page above
- Make sure you choose g++ for compiling the helloworld.cpp, I accidentally chose gcc first as the compiler and encountered a big list of errors!
- At the bottom of the page there is some information about how to use the debugger. I think it is a good idea to learn about debugging basics, this knowledge **will** save you a lot of time later!

[*] See Appendix A if you want to know what MinGW64, MSYS64 or UCRT64 mean.

Install the OpenCV Libraries

- Open an MSYS2 UCRT64 shell:
  - Search for “msys2” in the search window of the taskbar, choose MSYS2 UCRT64 and open one UCRT64 shell →



```
ahmet@MU00166820 UCRT64 ~
$ |
```

- In this shell, run:

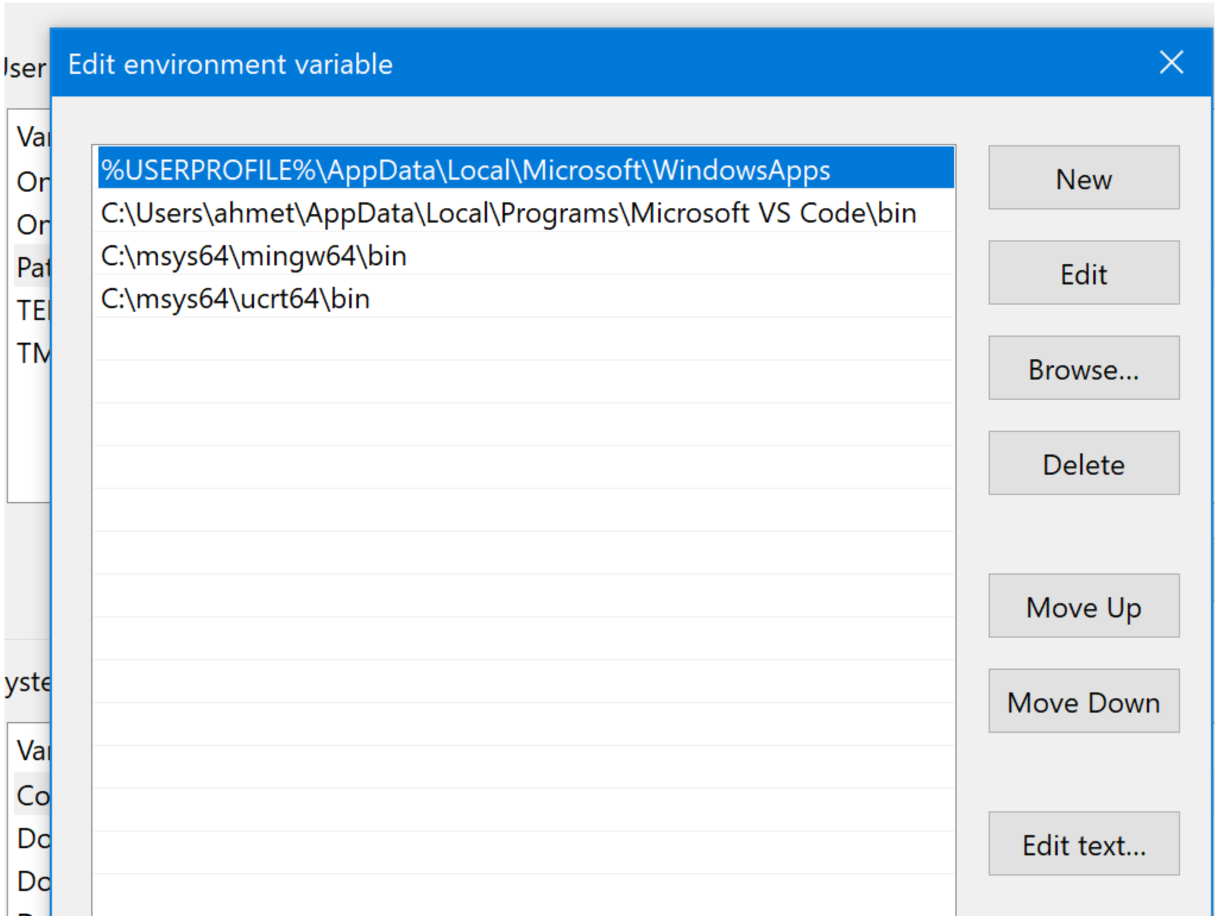
  1. pacman -S mingw-w64-x86_64-opencv
  2. pacman -S mingw-w64-x86_64-qt6-base

  to install the OpenCV libraries

Update the Path Environment Variable (Again!)

- In the Windows search bar, type Settings to open your Windows Settings
- Search for Edit environment variables for your account
- In your User variables, select the Path variable and then select Edit
- Select New and add C:\msys64\mingw64\bin
- Reorder the list to make sure that C:\msys64\mingw64\bin appears above C:\msys64\ucrt64\bin (see next slide)
- Select OK to save the updated Path. You will need to reopen any console windows for the new Path location to be available.

Directories in the Path Environment Variable: Order is Important


Edit environment variable

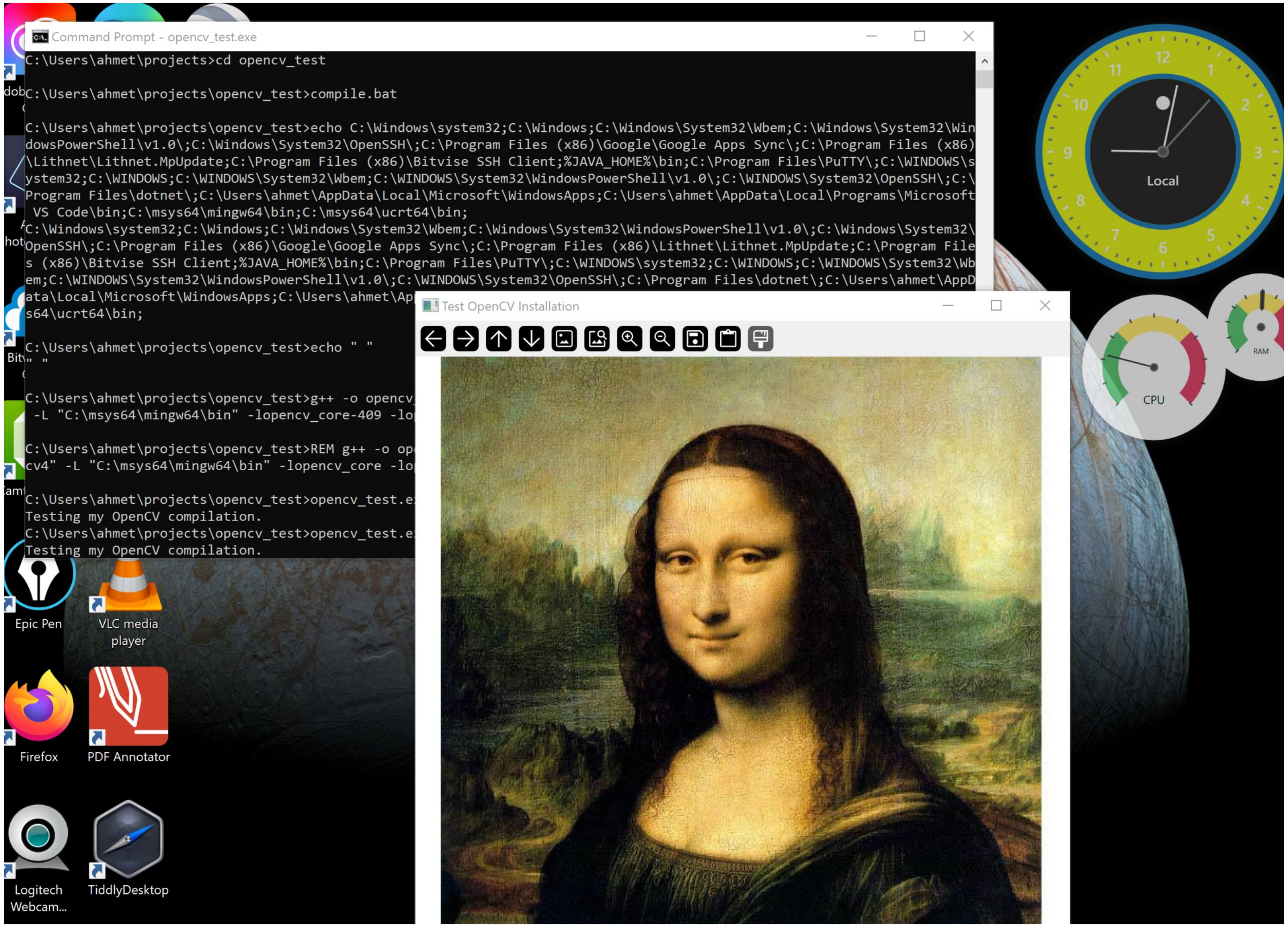
```
%USERPROFILE%\AppData\Local\Microsoft\WindowsApps
C:\Users\ahmet\AppData\Local\Programs\Microsoft VS Code\bin
C:\msys64\mingw64\bin
C:\msys64\ucrt64\bin
```

Test the OpenCV Libraries

- In the “projects” directory (where you created your helloworld program), create a directory “opencv_test” (in my case, the directory is: C:\Users\ahmet\projects\opencv_test)
- Copy these files into opencv_test:
  - opencv_test.cpp
  - mona_lisa.jpg
  - compile.bat
- Open a Windows Terminal: “cmd”, cd projects\opencv_test
- Type compile.bat at the command prompt to compile and link your program [*]
- Type opencv_test.exe at the command prompt to see Mona Lisa on the screen

[*] See Appendix B to know how the OpenCV programs are compiled in VSCode.

Test OpenCV Libraries



```
C:\Users\ahmet\projects>cd opencv_test

C:\Users\ahmet\projects\opencv_test>compile.bat

C:\Users\ahmet\projects\opencv_test>echo C:\Windows\system32;C:\Windows;C:\Windows\System32\Wbem;C:\Windows\System32\Win
dowsPowerShell\v1.0\;C:\Windows\System32\OpenSSH\;C:\Program Files (x86)\Google\Google Apps Sync\;C:\Program Files (x86)
\Lithnet\Lithnet.MpUpdate;C:\Program Files (x86)\Bitvise SSH Client;%JAVA_HOME%\bin;C:\Program Files\PuTTY\;C:\WINDOWS\s
ystem32;C:\WINDOWS;C:\WINDOWS\System32\Wbem;C:\WINDOWS\System32\WindowsPowerShell\v1.0\;C:\WINDOWS\System32\OpenSSH\;C:\
Program Files\dotnet\;C:\Users\ahmet\AppData\Local\Microsoft\WindowsApps;C:\Users\ahmet\AppData\Local\Programs\Microsoft
 VS Code\bin;C:\msys64\mingw64\bin;C:\msys64\ucrt64\bin;
C:\Windows\system32;C:\Windows;C:\Windows\System32\Wbem;C:\Windows\System32\WindowsPowerShell\v1.0\;C:\Windows\System32\
OpenSSH\;C:\Program Files (x86)\Google\Google Apps Sync\;C:\Program Files (x86)\Lithnet\Lithnet.MpUpdate;C:\Program File
s (x86)\Bitvise SSH Client;%JAVA_HOME%\bin;C:\Program Files\PuTTY\;C:\WINDOWS\system32;C:\WINDOWS;C:\WINDOWS\System32\Wb
em;C:\WINDOWS\System32\WindowsPowerShell\v1.0\;C:\WINDOWS\System32\OpenSSH\;C:\Program Files\dotnet\;C:\Users\ahmet\AppD
ata\Local\Microsoft\WindowsApps;C:\Users\ahmet\Ap
s64\ucrt64\bin;

C:\Users\ahmet\projects\opencv_test>echo " "
" "

C:\Users\ahmet\projects\opencv_test>g++ -o opencv
 -L "C:\msys64\mingw64\bin" -lopencv_core-409 -lo

C:\Users\ahmet\projects\opencv_test>REM g++ -o op
cv4" -L "C:\msys64\mingw64\bin" -lopencv_core -lo

C:\Users\ahmet\projects\opencv_test>opencv_test.e
Testing my OpenCV compilation.
C:\Users\ahmet\projects\opencv_test>opencv_test.e
Testing my OpenCV compilation.
```

Appendix A

- MSYS2: Software distribution and development platform for Windows
- UCRT64: Universal runtime library for 64-bit Windows
- MinGW-w64: Compiler suite for 64-bit Windows

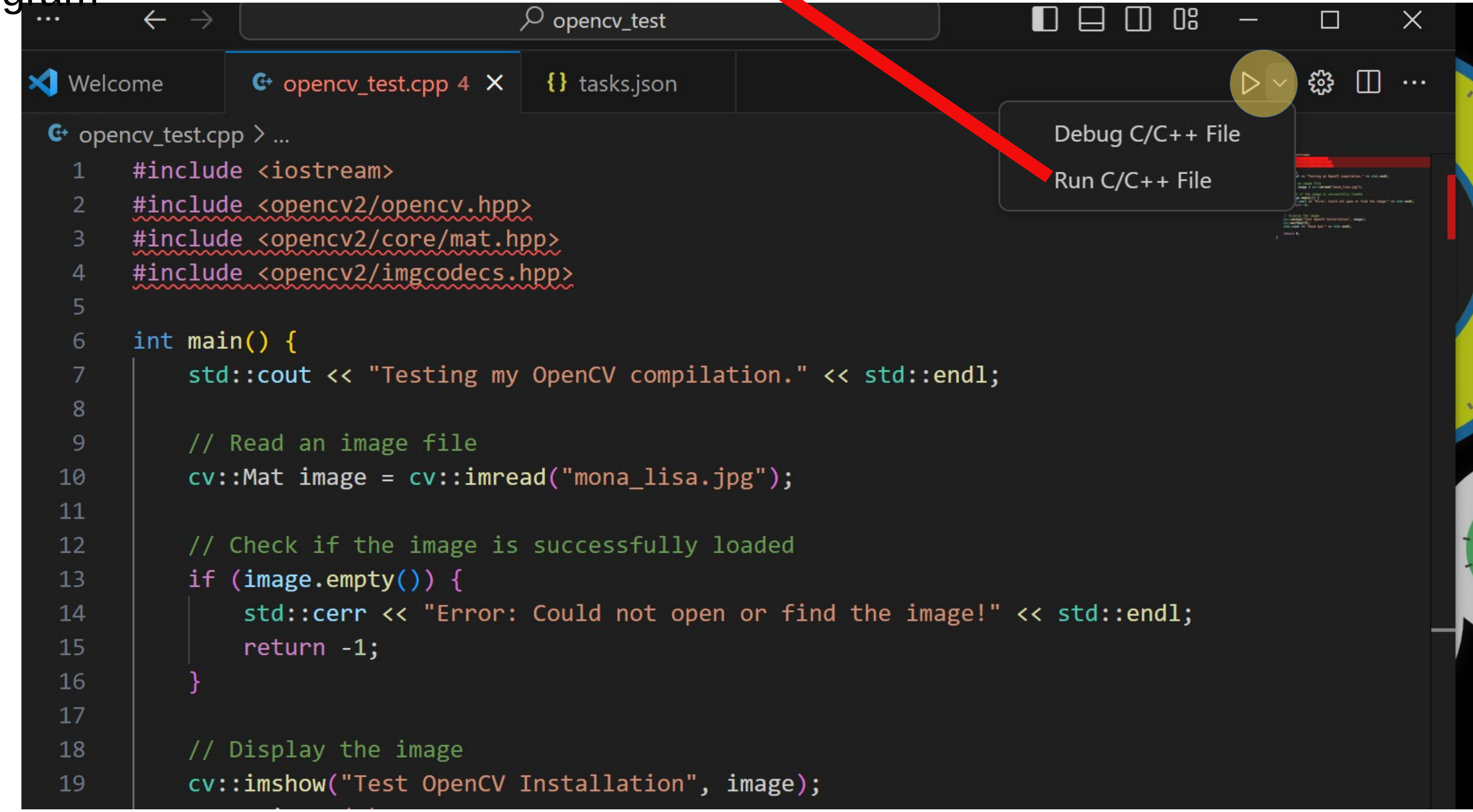
Appendix B: How to Compile OpenCV Programs in VSCode?

- I assume that you have successfully run the Mona Lisa program (slide 7).
- Open a Windows Terminal: “cmd”, cd projects\opencv_test
- run: code . (← don’t forget to type “.”)

Click the triangle at the upper right and “Run C/C++ File”, a menu will appear, choose g++
Compilation will terminate with errors, but this action will create a tasks.json file **under** the .vscode directory.

→ Replace this file with ours

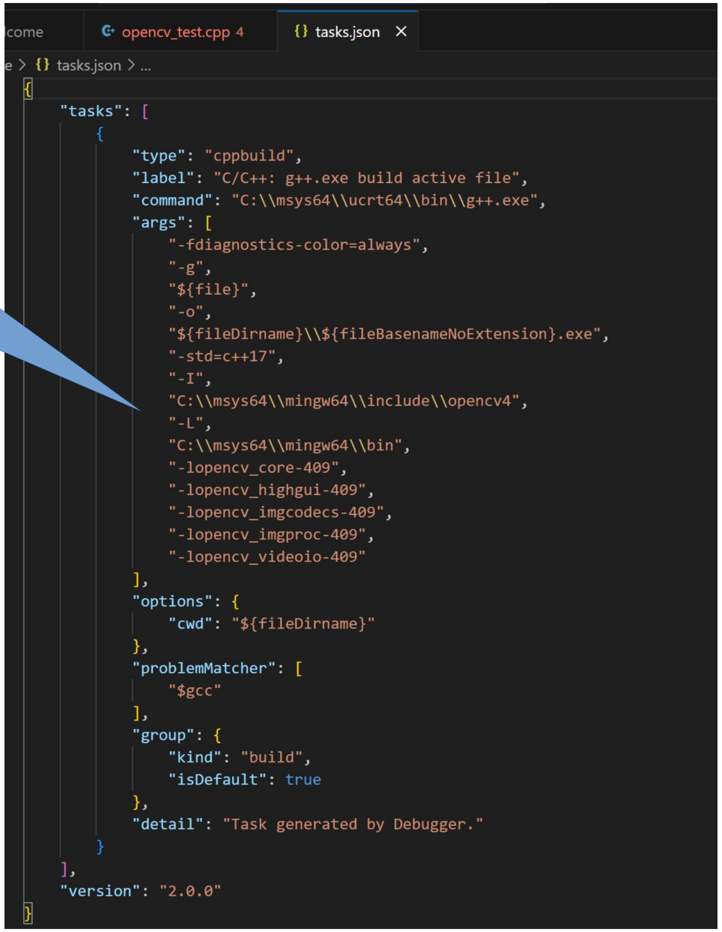
→ Recompile the program



```cpp
#include <iostream>
#include <opencv2/opencv.hpp>
#include <opencv2/core/mat.hpp>
#include <opencv2/imgcodecs.hpp>

int main() {
    std::cout << "Testing my OpenCV compilation." << std::endl;

    // Read an image file
    cv::Mat image = cv::imread("mona_lisa.jpg");

    // Check if the image is successfully loaded
    if (image.empty()) {
        std::cerr << "Error: Could not open or find the image!" << std::endl;
        return -1;
    }

    // Display the image
    cv::imshow("Test OpenCV Installation", image);
```

New tasks.json

Necessary command line arguments for the g++ compiler/linker



```
{
    "tasks": [
        {
            "type": "cppbuild",
            "label": "C/C++: g++.exe build active file",
            "command": "C:\\msys64\\ucrt64\\bin\\g++.exe",
            "args": [
                "-fdiagnostics-color=always",
                "-g",
                "${file}",
                "-o",
                "${fileDirname}\\${fileBasenameNoExtension}.exe",
                "-std=c++17",
                "-I",
                "C:\\msys64\\mingw64\\include\\opencv4",
                "-L",
                "C:\\msys64\\mingw64\\bin",
                "-lopencv_core-409",
                "-lopencv_highgui-409",
                "-lopencv_imgcodecs-409",
                "-lopencv_imgproc-409",
                "-lopencv_videoio-409"
            ],
            "options": {
                "cwd": "${fileDirname}"
            },
            "problemMatcher": [
                "$gcc"
            ],
            "group": {
                "kind": "build",
                "isDefault": true
            },
            "detail": "Task generated by Debugger."
        }
    ],
    "version": "2.0.0"
}
```