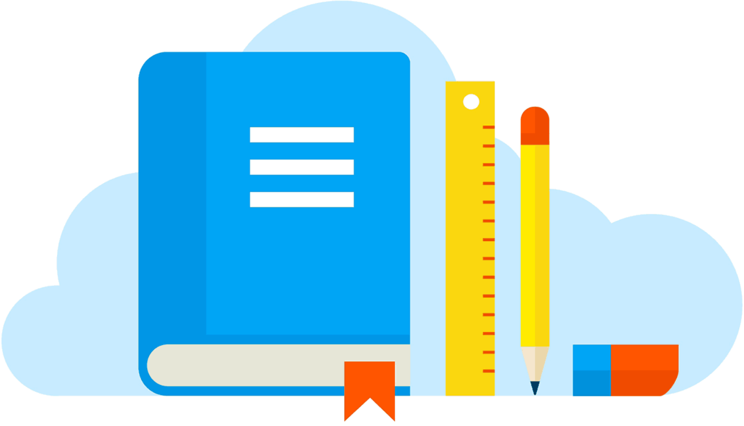


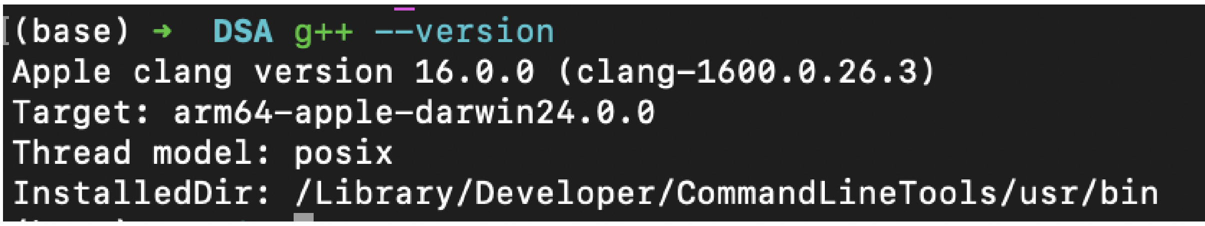
How to Run C/C++ Programs on VSCode

Date: 17/04/2025

In this document, `terminal` refers to Command Prompt on Windows, and Terminal on Mac and Linux.

1. Install a C++ Compiler

- Try to run command on terminal `g++ --version`



```
(base) ➜  DSA g++ --version
Apple clang version 16.0.0 (clang-1600.0.26.3)
Target: arm64-apple-darwin24.0.0
Thread model: posix
InstalledDir: /Library/Developer/CommandLineTools/usr/bin
```

If your terminal shows an output similar to the image above, it means that g++ is successfully installed on your system.

- If you haven't installed g++ yet, follow the instructions below based on your operating system.
  - Windows: install MinGW
  - Mac: run `xcode-select --install` on terminal
  - Linux: run `sudo apt install g++` on terminal

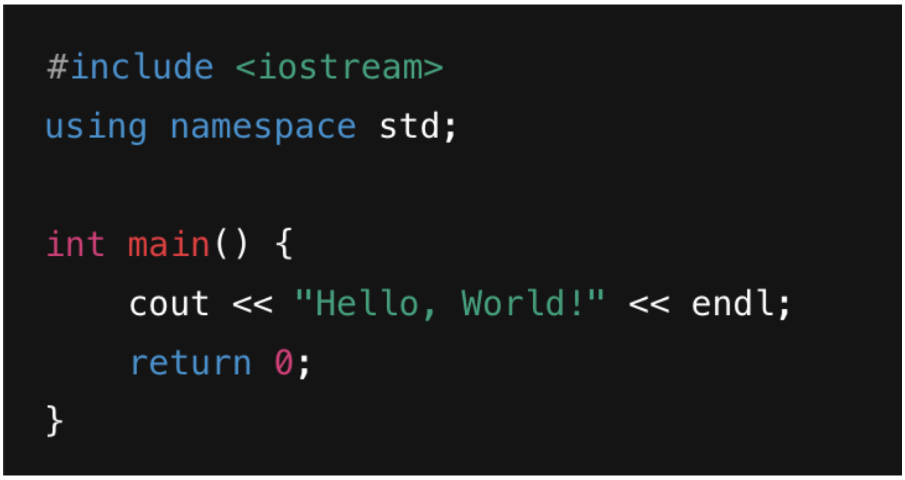
After installing, open Terminal and type: `g++ --version`

2. Install VSCode

If you don't have VSCode installed, you can download it from the following link: https://code.visualstudio.com/download.

3. Write a Simple C++ Program

- Open VSCode.
- Create a new file: `main.cpp`.
- Write the following code:



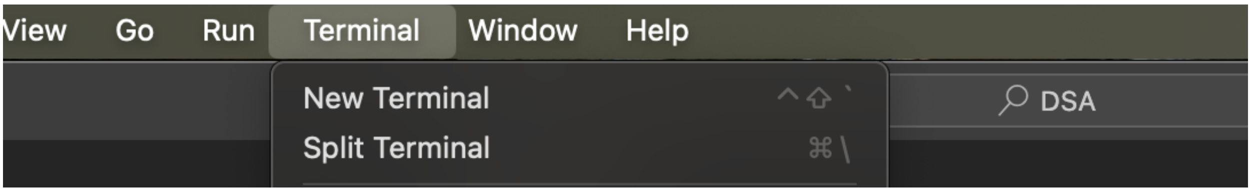
```cpp
#include <iostream>
using namespace std;

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

- Save the file (`Ctrl+S`).

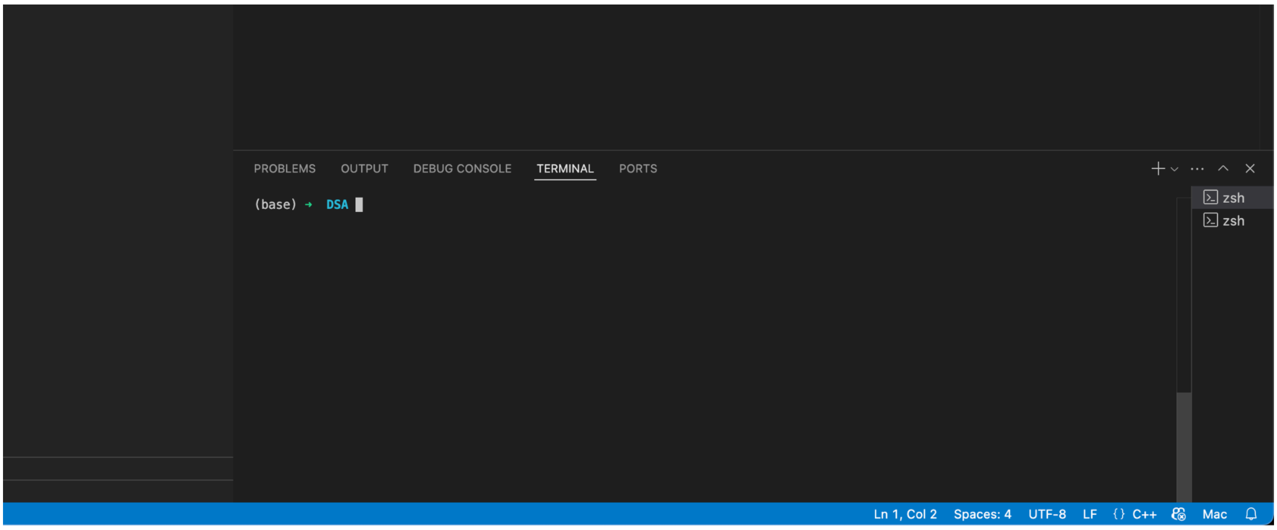
4. Build and Run Using Terminal

4.1 Open Terminal in VSCode

- Press `` Ctrl+` `` (the backtick key) or
- Go to **Terminal > New Terminal**.



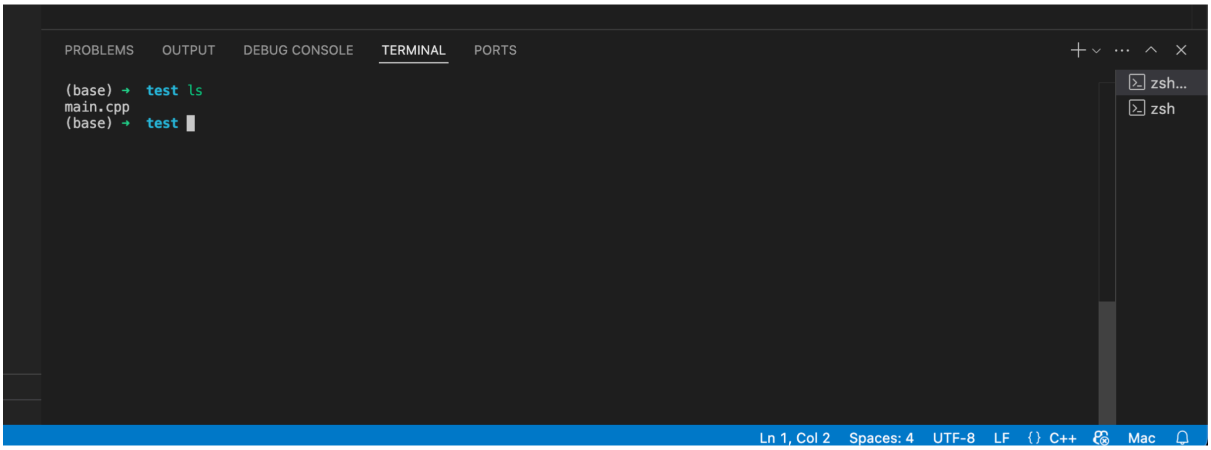
The terminal will open and appear at the bottom of the VSCode window.



```
(base) → DSA
```

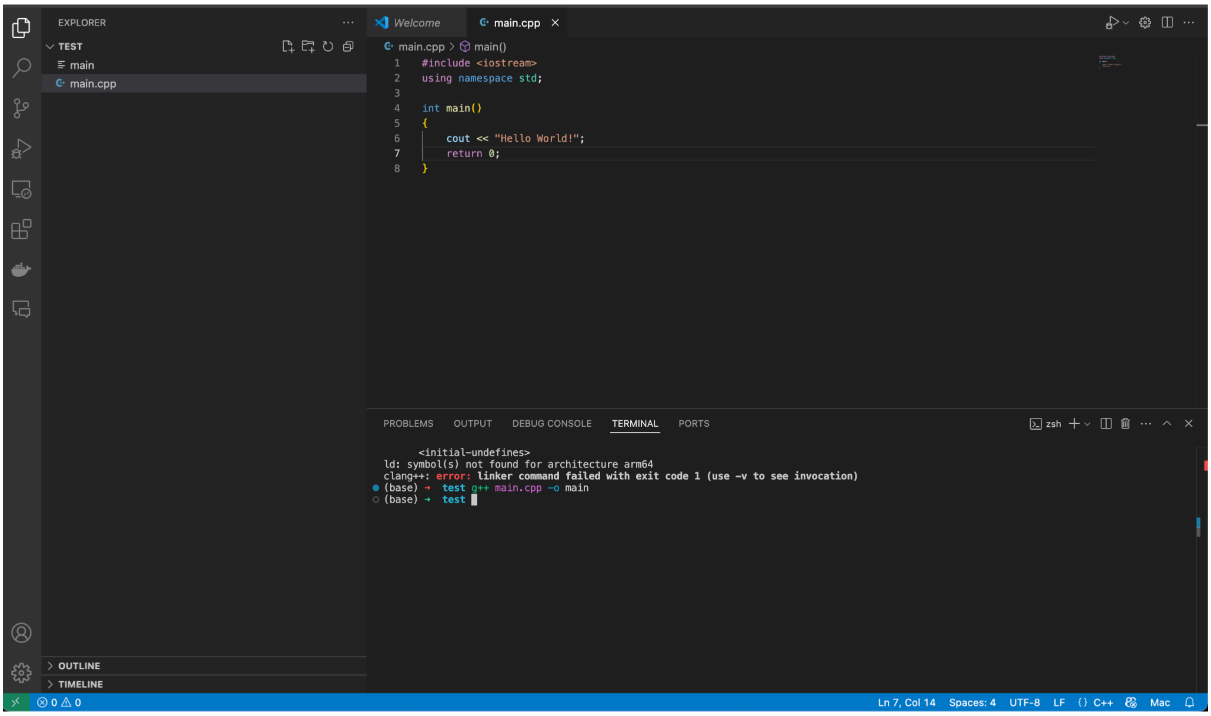
4.2 Compile the Program

- In the terminal, type: `g++ main.cpp -o main` (`-o main` tells the compiler to create an executable named main). Before running the command, make sure your terminal is already in the same directory as your `main.cpp` file. You can check your current files by typing `ls` (on Mac/Linux) or `dir` (on Windows).



```
(base) → test ls
main.cpp
(base) → test
```

- This will create a file named `main`, which is an executable file.



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

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

```
      <initial-undefines>
ld: symbol(s) not found for architecture arm64
clang++: error: linker command failed with exit code 1 (use -v to see invocation)
(base) ➜  test g++ main.cpp -o main
(base) ➜  test 
```

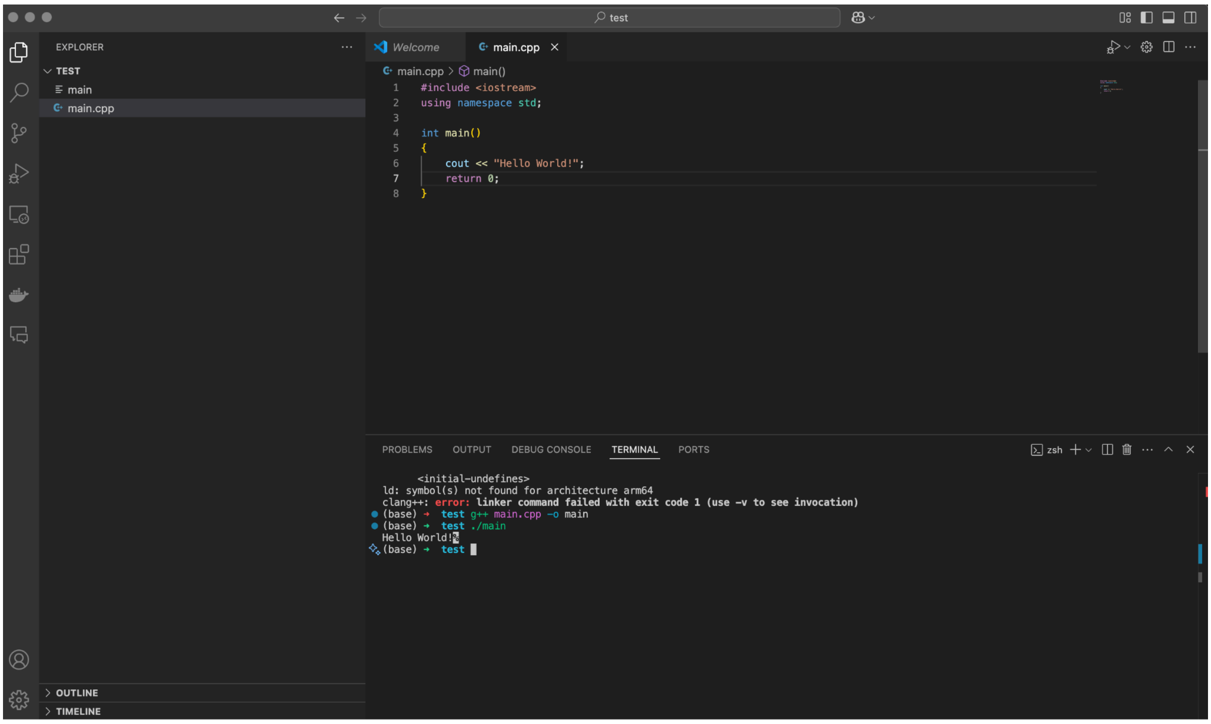
4.3 Compile a project with multiple files

To compile and build the project, we need to compile all the `.cpp` files together. Use the following command in the terminal:

```
g++ -std=c++11 *.cpp -o my_project
```

4.4 Run the Program

Run `./main` on terminal



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

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

```
      <initial-undefines>
ld: symbol(s) not found for architecture arm64
clang++: error: linker command failed with exit code 1 (use -v to see invocation)
(base) → test g++ main.cpp -o main
(base) → test ./main
Hello World!%
(base) → test
```

5. Common Mistakes to Watch For

- Make sure your terminal's current directory is where your `.cpp` file is saved.
- Check that g++ is installed and properly added to your system PATH.
- Use correct file names (case-sensitive on Mac/Linux).
- When you use features like `vector<int>` (or other modern C++ features), you should compile your program using the C++11 standard or later. You should use the command `g++ -std=c++11 main.cpp -o main`. This tells the compiler to use the C++11 standard. Without this, you might get errors when using vector, auto, for-each loops, and other modern C++ features.